Alma Mater Studiorum - Università di Bologna

DOTTORATO DI RICERCA IN

INGEGNERIA BIOMEDICA, ELETTRICA E DEI SISTEMI

Ciclo 35

Settore Concorsuale: 01/A6 - RICERCA OPERATIVA

Settore Scientifico Disciplinare: MAT/09 - RICERCA OPERATIVA

ADJUSTABLE ROBUST OPTIMIZATION WITH NONLINEAR RECOURSES

Presentata da: Henri Bertrand Roger Jean-Marc Arthur Lefebvre

Coordinatore Dottorato

Michele Monaci

Supervisore

Michele Monaci

Co-supervisore

Enrico Malaguti

Esame finale anno 2023

\We should not speak so that it is possible for the audience to understand us, but so that it is impossible for them to misunderstand us."

—M. F. Quintiliano, *De Institutione Oratoria* (c. 95 AD)

Acknowledgements

First and foremost, I would like to address my deepest thanks to Pr. Michele Monaci and to Pr. Enrico Malaguti, who were my Ph.D advisors. I am particularly grateful for the trust they granted me throughout these years, and the mature relationship we could build. Their advice and guidelines are inestimable.

I am also grateful to Pr. Boris Detienne with whom working has always been a pleasure.

To Pr. François Clautiaux, I would like to address a warm thank you for the work we have conducted together, but also for his constant availability in times of doubt.

Many thanks should go to my colleagues from the DEI laborartory of the University of Bologna for making these years an enjoyable moment. In particular, I would like to thank Naga Venkata C. Gudapati for his kindness.

I would also like to acknowledge my dear colleagues from the MINOA project, with special mentions to Benedikt Bienhüls and Dennis Adelhütte.

The first two years of my research were supported by the MINOA consortium, which received funding from the European Union's EU Framework Programme for Research and Innovation Horizon 2020 under the Marie Sklodowska-Curie Actions Grant Agreement No 764759. I am grateful for the opportunities they offered me.

The last year of my Ph.D was supported by the Air Force Office of Scientific Research under award FA8655-20-1-7019. My research would not have been possible without this opportunity.

To Charlotte, thank you for all the rest.

Abstract

Over the last century, mathematical optimization has become a prominent tool for decision making. Its systematic application in practical fields such as economics, logistics or defence led to the development of algorithmic methods with ever increasing efficiency. Indeed, for a variety of real-world problems, finding an optimal decision among a set of (implicitly or explicitly) predefined alternatives has become conceivable in reasonable time. In the last decades, however, the research community raised more and more attention to the role of uncertainty in the optimization process. In particular, one may question the notion of optimality, and even feasibility, when studying decision problems with unknown or imprecise input parameters. This concern is even more critical in a world becoming more and more complex —by which we intend, interconnected —where each individual variation inside a system inevitably causes other variations in the system itself.

In this dissertation, we study a class of optimization problems which suffer from imprecise input data and feature a two-stage decision process, i.e., where decisions are made in a sequential order —called stages —and where unknown parameters are revealed throughout the stages. The applications of such problems are plethora in practical fields such as, e.g., facility location problems with uncertain demands, transportation problems with uncertain costs or scheduling under uncertain processing times.

In the first chapter of this thesis, we present the main existing approaches for dealing with uncertain parameters in mathematical optimization along with their underlying assumptions. In particular, we discuss the main motivations behind the robust optimization paradigm, to which this thesis aims at contributing. Chapter 2 is then dedicated to a deeper introduction to robust optimization and its existing solution approaches. In this chapter, it is also made clear that further effort is needed for a sub-class of problems where second-stage decisions are nonlinear (e.g., mixed-integer and/or convex). The second part of the thesis is then dedicated to original contributions to the two-stage robust optimization field, and is organised as follows.

– In Chapter 3, we study problems in which costs are subject to uncertainty and the second-stage decisions are modeled as a mixed-integer nonlinear problem (MINLP). Extending a recent contribution from the literature, we introduce a new general approach for our class of problems. *This work is under revision at EJOR - European Journal of Operational Research.*

– In Chapter 4, we apply the previous method to a practical problem arising in scheduling where jobs are subject to failure. *This work has been published in Journal of Scheduling (see Clautiaux et al. [2023]).*

– In Chapter 5, we study problems where the uncertain parameters are binary and the second stage mixed-integer. We introduce an enumerative algorithm akin to a Benders branch-and-cut scheme able to solve this class of problems. *This work is under revision at INFORMS Journal on Computing.*

– In Chapter 6, we apply the previous method to a facility location problem with unknown demands. *This work was accepted as a conference paper at ROADEF 2022 which took place in Lyon (France), in 2022.*

– In Chapter 7, we study problems where the second-stage decisions are defined by means of general convex constraints and show that such problems can be solved by column-and-constraint generation when the set of possible scenarios has an affine mapping to a 0 1 polytope.

The thesis ends with a conclusion and future research directions.

Contents

List of Figures

List of Tables

List of Notations

Linear Algebra

$\boldsymbol{x} \in \mathbb{R}^n$	n-dimensional real vector
$\boldsymbol{A} \in \mathbb{R}^{m \times n}$	real-valued matrix with m rows and n columns
$\boldsymbol{a}^{(j)} \in \mathbb{R}^m$	jth column of matrix $\boldsymbol{A}$
$\boldsymbol{a}_{(i)} \in \mathbb{R}^{1 \times n}$	ith row of matrix $\boldsymbol{A}$
$a_{ij} \in \mathbb{R}$	component (i,j) of matrix $\boldsymbol{A}$
$\boldsymbol{A}^T$	transpose of $\boldsymbol{A}$
$\|\boldsymbol{x}\|_p$	ℓ_p-norm of $\boldsymbol{x}$ ($p \in \mathbb{N} \cup \{+1\}$)
$\text{diag}(\boldsymbol{A})$	diagonal of matrix $\boldsymbol{A}$ as a column vector
$\text{diag}(\boldsymbol{a}^{(j)})$	(n,n)-dimensional diagonal matrix whose (j,j) component is a_j ($j = 1, ..., n$)
$\{\boldsymbol{x}^p\}_{p \in P} \to \bar{\boldsymbol{x}}$	convergent sequence indexed by $p \in P$ with limit $\bar{\boldsymbol{x}}$

Convex Optimization

$\overline{\mathbb{R}}$	extended real line, i.e., $\overline{\mathbb{R}} = \mathbb{R} \cup \{-1, +1\}$
$f : \mathbb{R}^n \to \overline{\mathbb{R}}$	multivariate function from $\mathbb{R}^n$ to $\overline{\mathbb{R}}$
$\text{dom}\,(f)$	domain of function f, i.e., $\text{dom}\,(f) = \{\boldsymbol{x} \in \mathbb{R}^n : f(\boldsymbol{x}) < +1\}$
$f(X)$	image of f over X (with $X \subseteq \text{dom}\,(f)$)
$f^* : \mathbb{R}^n \to \mathbb{R}$	convex conjugate of function f, i.e., $f^*(\boldsymbol{\pi}) = \sup\{\boldsymbol{\pi}^T\boldsymbol{x} \quad f(\boldsymbol{x}) : \boldsymbol{x} \in \text{dom}\,(f)\}$
$f_* : \mathbb{R}^n \to \mathbb{R}$	concave conjugate of function f, i.e., $f_*(\boldsymbol{\pi}) = \inf\{\boldsymbol{\pi}^T\boldsymbol{x} \quad f(\boldsymbol{x}) : \boldsymbol{x} \in \text{dom}\,(\ ) f\}$
$\psi(\ , \boldsymbol{y})$	equivalent to $\boldsymbol{x} \mapsto \psi(\boldsymbol{x}, \boldsymbol{y})$ for a fixed $\boldsymbol{y}$ (when appropriate)
$\delta(\ \|X)$	indicator function of set X, i.e., evaluates to $+1$ if its argument is in X, 0 otherwise
$\partial f(\boldsymbol{x})$	set of sub-gradients of f at $\boldsymbol{x}$
$\text{cont}\,(X)$	continuous relaxation of set X (when context is clear)
$\text{int}(X)$	interior of set X
$\text{conv}\,(X)$	convex hull of set X
$\text{vert}\,(X)$	set of extreme points of the convex polyhedron X
$\text{proj}_{\boldsymbol{y}}(C)$	projection of C onto the $\boldsymbol{y}$ variables (when context is clear)

Probability Theory

$(\widehat{\ }, F, \mathsf{P})$	probability space
$\mathsf{P}(X)$	probability of a random event X
$\mathsf{E}(\varphi(X))$	expected value of $\varphi(X)$ with X a random event and $\varphi : F \ ! \ \mathsf{R}$
$N(\mu, \sigma^2)$	normal law of mean μ and variance σ^2
$N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	multivariate normal law of mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$
$U(a, b)$	continuous uniform law between a and b

Logic Theory

$\mathbb{1}(E)$	function which equals 1 if E is true, 0 otherwise
$:E$	logical negation of proposition E
$E_1 \wedge E_2$	logical "and" between propositions E_1 and E_2
$E_1 \_ E_2$	logical "or" between propositions E_1 and E_2

Others

N	set of natural numbers
N	set of natural numbers strictly greater than 0
Z	set of integer numbers

Part I

Introduction

CHAPTER 1

Decision making under uncertainty

1.1 Introduction

What is a *good* decision? Decision making is a complex subject. In Simon [1993], the American political scientist Herbert A. Simon describes three types of decisions: *irrational*, those which are poorly adapted to one's goal; *nonrational*, those for which the goal cannot clearly be identified and, finally, *rational*, those which lead to well adapted actions with respect to one's goal. While the first two kinds of decisions are of great interest, even computationally, we will not address them. In this thesis, we will always refer to decision making as *rational* decision making.

As we understand it, decision making considers an alternative of choice consisting in, at least, two different options and aims at selecting one of them so as to reach a given objective. Thus let X be a set of feasible decisions (also called *solutions*) and ψ be a function, defined over X, associating a score to each solution. Informally speaking, ψ (which we refer to as the *objective function*) is a measure of how *good* a decision is and is to be understood as "the lower $\psi(\boldsymbol{x})$, the better decision $\boldsymbol{x}$ is". When the image of ψ, noted $\psi(X)$, lies in a partially ordered set, one is concerned by a *multi-objective* problem. An interested reader may refer to Mandal et al. [2018] for a complete introduction. In this thesis, however, we will always assume that $\psi(X)$ lies in a totally ordered set. Even more, that $X \subseteq \mathsf{R}^{n_X}$ (for some fixed $n_X \in \mathsf{N}$) and $\psi : X \to \overline{\mathsf{R}}$. Thus, given two decisions $\boldsymbol{x}$ and $\boldsymbol{x}'$ in X, we have that "$\boldsymbol{x}$ is a better decision than $\boldsymbol{x}'$ if and only if $\psi(\boldsymbol{x}) < \psi(\boldsymbol{x}')$". Thus, given the objective function ψ, one is interested in minimizing function ψ over the domain X, which we write as

$$\inf_{\boldsymbol{x} \in X} \psi(\boldsymbol{x}). \tag{1.1}$$

Problem (1.1), without any further restricting assumption, is impossible to solve computationally. This is trivial given that there exists uncomputable functions (see Turing [1937]). This might seem a technically, but it is not. Indeed, a careful reader may have noticed that, so far, we have not really answered the very first question that was asked: What is a *good* decision? Sure, a

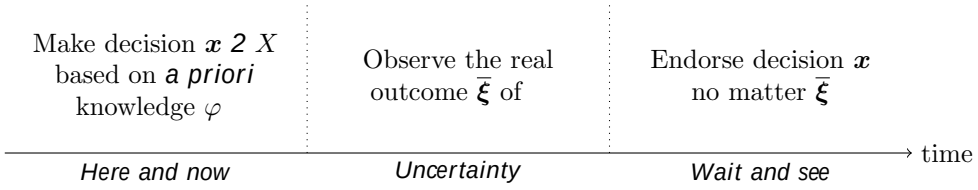


Figure 1.1: Single-stage decision flow under uncertainty

good decision is one that minimizes ψ, but what does ψ look like? How do we know? Let me be very specific: is the function ψ which returns 0 if its argument is the *best* decision to make and 1 otherwise an interesting choice for ψ? In theory, it sure is. In practice, however, it is useless.

Defining the shape of ψ and X is what is called *modeling* a problem. It consists in studying a real-world problem and deriving mathematical expressions which are *thought* to describe it *well enough*. Every model is subjective. This is a fact, and not much can be done about it. However, it is very common that ψ (alternatively, X) is defined in terms of some parameters which have to be measured or estimated. For instance, consider a routing problem where someone must decide a route from a city A to B. A *reasonable* model for this problem would be to consider every feasible route from A to B which cost less than a given budget and to select the one minimizing the total travel time. Here, X is the set of feasible routes linking A to B respecting the budget while, given a route $\boldsymbol{x} \in X$, $\psi(\boldsymbol{x})$ returns the total travel time of $\boldsymbol{x}$. This model may readily be cast into (1.1). Yet, it is easily seen that the objective function, i.e., the travel time of each route, must be estimated and will depend on multiple parameters such as the weather, the time of day or even very specific scenarios such as social strikes and road closings. Thus, in reality, ψ may be expressed in terms of external parameters. Let $\boldsymbol{\xi} \in \mathbb{R}^{n_\Xi}$ be a vector of such parameters (for a fixed $n_\Xi \in \mathbb{N}$), to better account for the dependence of ψ on $\boldsymbol{\xi}$, we will write $\psi(\boldsymbol{x}) = \psi(\boldsymbol{x}; \boldsymbol{\xi})$. Now, assume that the actual value of $\boldsymbol{\xi}$ can be observed before the decision time, i.e., here and now, and takes value $\overline{\boldsymbol{\xi}}$. Then, (1.1) is readily solved by the following problem.

$$\inf_{\boldsymbol{x} \in X} \psi(\boldsymbol{x}; \overline{\boldsymbol{\xi}}) \tag{1.2}$$

In practice, assuming that those parameters $\boldsymbol{\xi}$ can be observed here and now, and with an infinite precision, is very optimistic. Consider planning a trip in one months, can the weather be estimated well enough? Can the social strikes be easily foreseen? In other words, can $\boldsymbol{\xi}$, and thus $\psi(\ ; \boldsymbol{\xi})$, be estimated well enough? This, off course, depends on the situation, but it typically is not the case. In fact, one may rather see the sought parameters as realizations of random events. This consideration is the starting point of what is referred to as *optimization under uncertainty*.

More formally, let $\quad$ be an n_Ξ-dimensional random vector in the probability space $(\widehat{\ }, \mathcal{F}, \mathbb{P})$ where $\widehat{\ }$ is the set of all possible outcomes for $\quad$, $\mathcal{F}$ is an event space (set of subsets of outcomes) and $\mathbb{P}$ is a probability measure which assigns a probability to each event in $\mathcal{F}$. We introduce function $\varphi \quad : (X \quad \widehat{\Xi} \to \mathbb{R}) \quad X \to \mathbb{R}$ which estimates the impact of a decision onto the objective function with respect to $\quad$. For instance, $\varphi \quad$ may no longer return "the" travel time of a route, but well the *expected* travel time of a route (i.e., the average travel time for the route over an infinite test bed). In this case, we have $\varphi \ (\psi, \boldsymbol{x}) = \mathbb{E}\,(\psi(\boldsymbol{x}; \quad))$.

Figure 1.1 depicts the actual decision flow which we consider here. Clearly, here and now,

a decision $\boldsymbol{x} \in X$ must be taken. However, the outcome $\boldsymbol{\xi}$ of cannot yet be observed, and is therefore uncertain. Instead, one may take decision $\boldsymbol{x}$ based on the *a priori* knowledge of as measured by φ . Later, the actual outcome $\overline{\boldsymbol{\xi}}$ of will be observed and the taken decision $\boldsymbol{x}$ will have to be endorsed. Thus, the decision maker's goal, here and now, is to decide $\boldsymbol{x}$ so that, in the future, the taken decision is regarded as *good*. With an *appropriate* a priori knowledge on , φ , we thus aim at solving the following problem.

$$\inf_{\boldsymbol{x} \in X} \varphi\ (\psi, \boldsymbol{x}) \tag{1.3}$$

Again, what is meant by "an *appropriate* a priori knowledge on " depends on the situation. In the next section, we will present four widely studied approaches for designing φ . Informally speaking, *chance-constrained optimization* aims at controlling the probability of taking a *good* decision while *stochastic optimization* seeks at taking *averagely good* decisions. *Robust optimization* looks for *good* decisions in the *worst* scenario while *distributionally robust optimization* can be seen as a combination of stochastic and robust optimization. All four approaches will be introduced in light of the current discussion, i.e., for each paradigm, ψ and X will be formally given so as to fit in (1.3). When needed, some more natural formulations will also be presented in order to agree with the scientific literature and to reach more readability. Indeed, all three fields have their own convention and notational habits, which we intend to follow in the rest of the dissertation.

We end this section by formally defining what we refer to as a *(parametrized) optimization model* and its uncertain counterpart.

Definition 1 (Parametrized optimization model)**.** *Let* $\widehat{\Xi} \subseteq \mathsf{R}^{n_\Xi}$ *and* $X \subseteq \mathsf{R}^{n_X}$ *be two given sets and let* $\psi : X \times \widehat{\Xi} \to \mathsf{R}$*. Then,* $(\psi, X, \widehat{\Xi})$ *is called a* (parametrized) optimization model*. The optimization problems associated to this model is written as*

$$\left\{ \inf_{\boldsymbol{x} \in X} \psi(\boldsymbol{x}; \boldsymbol{\xi}) \right\}_{\boldsymbol{\xi} \in \widehat{\Xi}}. \tag{1.4}$$

Remark 1 (Constraint uncertainty)**.** *In this section, we have assumed that only* ψ *was parametrized by unknown parameters. We briefly enlight that this is done without loss of generality. Indeed, assume that the feasible set* X *depends on some parameters* $\boldsymbol{\xi}$ *so that it may be written as* $X(\boldsymbol{\xi})$*. Then, one can build a model* (ψ^0, X^0) *accounting for it which can be cast as* (1.2)*. Indeed, we choose* $X^0 = \mathsf{R}^{n_X}$ *and* $\psi^0(\ ; \boldsymbol{\xi}) = \psi(\ ; \boldsymbol{\xi}) + \delta(\ | X(\boldsymbol{\xi}))$*, where* $\delta(\ | X(\boldsymbol{\xi}))$ *denotes the indicator function of set* $X(\boldsymbol{\xi})$*.*

Definition 2 (Uncertain optimization model)**.** *Let* $(\psi, X, \widehat{\Xi})$ *be a given optimization model and let* $\varphi : (X \times \widehat{\Xi} \to \mathsf{R}) \times X \to \mathsf{R}$ *be a given risk measure. Then,* $(\psi, X, \widehat{\Xi}, \varphi)$ *is called an* uncertain optimization model *and its associated optimization problem is 1.3.*

1.2 Mathematical modeling

1.2.1 Stochastic optimization

Stochastic Optimization (SO) finds its origin in the 1950s with an article by George B. Dantzig, entitled "Linear programming under uncertainty" (see Dantzig [1955]). Intuitively, a *reasonable*

decision, under uncertainty, could be defined as a decision which is optimal with respect to what is *reasonably expected* to happen. Thus, the stochastic optimization framework suggests to define φ in terms of the mathematical expectation. This is done as follows

$$\varphi\ (\psi, \boldsymbol{x}) = \mathsf{E}\,(\psi(\boldsymbol{x};\ \ )) = \int_{\boldsymbol{\xi} 2 \widehat{\Xi}} \psi(\boldsymbol{x}; \boldsymbol{\xi}) d\mathsf{P}. \tag{1.5}$$

Clearly, if ψ is a linear function of the unknown parameters, i.e., if there exists $\boldsymbol{\phi}$ such that $\psi(\boldsymbol{x}; \boldsymbol{\xi}) = \boldsymbol{\phi}(\boldsymbol{x})^T \boldsymbol{\xi}$, the obtained stochastic model is equivalent to take the expected value of the parameters and to use $\varphi\ (\boldsymbol{x}) = \psi(\boldsymbol{x}, \mathsf{E}\,(\ \ )) = \boldsymbol{\phi}(\boldsymbol{x})^T \mathsf{E}\,(\ \ )$. This fact directly follows from the linearity of expectation. However, when ψ fails to be linear, this may no longer be true. Indeed, let us assume that ψ is convex, then the following inequality, called Jensen's inequality, holds

$$\psi(\boldsymbol{x}; \mathsf{E}\,(\ \ ))\quad \mathsf{E}\,(\psi(\boldsymbol{x};\ \ ))\,. \tag{1.6}$$

In other words, replacing the parameters with their expected value, directly inside ψ, may lead to overly optimistic decisions.

Extreme care should be taken at this point. Inequality (1.6) would suggest that the deterministic model leads to better decisions, in the sense that its optimal solutions have a lower objective value than those arising from the stochastic model. In other words, we always have

$$\inf_{\boldsymbol{x} 2 X} \psi(\boldsymbol{x}; \mathsf{E}\,(\ \ ))\quad \inf_{\boldsymbol{x} 2 X} \mathsf{E}\,(\psi(\boldsymbol{x};\ \ ))\,. \tag{1.7}$$

However, recall that what matters here is the taken decision with respect to one's goal and its *adequacy* with the real world. As such, one should always remember that optimality is relative to a given criterion. Indeed, let $\boldsymbol{x}_{DT}$ be the optimal solution of the deterministic model (left hand-side) and let $\boldsymbol{x}_{SO}$ be the optimal solution of the stochastic model (right hand-side). By definition, we have that $\boldsymbol{x}_{DT}$ is a better decision than $\boldsymbol{x}_{SO}$, assuming that takes value $\mathsf{E}\,(\ \ )$, i.e., $\psi(\boldsymbol{x}_{DT}; \mathsf{E}\,(\ \ ))\quad \psi(\boldsymbol{x}_{SO}; \mathsf{E}\,(\ \ ))$. But we also have that $\boldsymbol{x}_{SO}$ is a better decision than $\boldsymbol{x}_{DT}$, assuming we know its probability distribution and the number of experiments is large enough, i.e., $\mathsf{E}\,(\psi(\boldsymbol{x}_{DT};\ \ ))\quad \mathsf{E}\,(\psi(\boldsymbol{x}_{SO};\ \ ))$. All in all, the following holds

$$\psi(\boldsymbol{x}_{DT}; \mathsf{E}\,(\ \ ))\quad \mathsf{E}\,(\psi(\boldsymbol{x}_{SO};\ \ ))\quad \mathsf{E}\,(\psi(\boldsymbol{x}_{DT};\ \ ))\,. \tag{1.8}$$

Clearly, when $\mathsf{E}\,(\psi(\boldsymbol{x}_{SO};\ \ ))\quad \psi(\boldsymbol{x}_{DT}; \mathsf{E}\,(\ \ ))$ is large, stochastic optimization is well justified.

For a complete introduction to the field of stochastic optimization, the reader may refer to Schneider and Kirkpatrick [2010].

1.2.2 Chance-constrained optimization

In the stochastic optimization framework, it is assumed that every outcome of should be taken into account when deciding $\boldsymbol{x}$. In that sense, the taken decision is always such that, for all possible outcome $\boldsymbol{\xi}$ of , we have $\psi(\boldsymbol{x}; \boldsymbol{\xi}) < +\infty$ (since, otherwise, we have $\mathsf{E}\,(\psi(\boldsymbol{x};\ \ )) = +\infty$). The Chance-Constrained Optimization (CCO) framework relaxes this assumption by ensuring that "$\psi(\boldsymbol{x};\ \ ) < +\infty$" is not violated up to a certain probability threshold. This is justified by the fact that, in many applications, dramatic scenarios do exist but with a very low probability. Thus, it

often is judicious to remove this tail when making a decision.

For a given decision $\boldsymbol{x} \in X$, let us define $P(\boldsymbol{x})$ as the random event "$\psi(\boldsymbol{x}; \ ) < +\infty$" and let F_α be the subset of X such that $\boldsymbol{x} \in F_\alpha$ if and only if the probability of $P(\boldsymbol{x})$ is greater than $1 - \alpha$, where $\alpha \in [0, 1)$ is an input parameter. Then, the CCO approach considers the following objective function

$$\varphi(\psi, \boldsymbol{x}) = \mathsf{P}(P(\boldsymbol{x}))\,\mathsf{E}(\psi(\boldsymbol{x}; \boldsymbol{\xi}) | P(\boldsymbol{x})) + \delta(\boldsymbol{x} | F_\alpha). \tag{1.9}$$

First, the term "$\delta(\boldsymbol{x}|F_\alpha)$" disqualifies any decision $\boldsymbol{x} \in X$ which would be such that $\mathsf{P}(\psi(\boldsymbol{x}; \ ) < +\infty) \quad 1 - \alpha$ by penalizing them with an infinite objective value. Then, the remaining term computes the expected value of $\psi(\boldsymbol{x}; \ )$, only on those outcomes $\boldsymbol{\xi}$ which fulfill $\psi(\boldsymbol{x}; \boldsymbol{\xi}) < +\infty$.

Remark 2 (Relation with stochastic optimization)**.** *Note that the CCO model* (1.9) *can be reduced to the SO model* (1.5) *by choosing* $\alpha = 0$*. In this case, we have* $\mathsf{P}(P(\boldsymbol{x})) = 1$ *and* $\mathsf{E}(\psi(\boldsymbol{x}; \boldsymbol{\xi}) | P(\boldsymbol{x})) = \mathsf{E}(\psi(\boldsymbol{x}; \boldsymbol{\xi}))$ *while* $\delta(\boldsymbol{x}|F_0)$ *becomes redundant.*

We now give the example of the Knapsack Problem with uncertain data.

Example 1 (Knapsack Problem with uncertain weight)**.** *The Knapsack Problem (KP) is a famous combinatorial problem where one must decide, among a set of items, a subset of items to put inside a container of limited capacity. Let* B *be this capacity and let* I *be a set of items. For each item* $i \in I$*, we denote its weight (i.e., the resource consumption of* i*) by* w_i *and denote its profit by* p_i *(i.e., the reward one gains by taking the item inside the container).*

This problem may be modeled by introducing, for each item $i \in I$*, a binary variable* x_i *which equals* 1 *if and only if* i *is put in the container and* 0 *otherwise. Then, the deterministic problem 1.1 takes the following form.*

$$\max \sum_{i \in I} p_i x_i \tag{1.10}$$

$$\text{s.t.} \sum_{i \in I} w_i x_i \quad B \tag{1.11}$$

$$x_i \in \{0, 1\} \qquad \forall i \in I \tag{1.12}$$

First, the objective (1.10) *maximizes the profit associated to the items selected in the container. Constraint* (1.11) *enforces that the container's capacity is not exceeded while* (1.12) *defines the domain of the* $\boldsymbol{x}$ *variables.*

Now, assume that the weight of each item $i \in I$ *depends, in reality, on external parameters and let us denote its weight by* $w_i(\overline{\boldsymbol{\xi}})$ *instead of* w_i*, where* $\overline{\boldsymbol{\xi}}$ *is an observed value for* . *Assuming that the probability distribution of* *is known, the chance-constrained counterpart of* (1.10)-(1.12) *is obtained by replacing* (1.11) *with the following constraint.*

$$\mathsf{P}\left(\sum_{i \in I} w_i(\ ) x_i \quad B\right) \quad 1 - \alpha \tag{1.13}$$

Note that, in this case, the objective function does not change. This is due to the fact that we did not assume that p_i *depends on* .

1.2.3 Robust optimization

A common assumption to stochastic and chance-constrained optimization is that one knows the probability distribution of the external parameters . For instance, this assumption is necessary for computing the expected value of $\psi(\boldsymbol{x};\ )$ for all $\boldsymbol{x}\ 2\ X$. This is a rather strong assumption in practice since one may not have such an *a priori* knowledge at hand. Moreover, even under this assumption, effectively computing $\mathsf{E}\,(\psi(\boldsymbol{x};\ ))$ for a given $\boldsymbol{x}\ 2\ X$ may be hard in practice and result in an NP-hard uncertain counterpart —even when the underlying deterministic problem is polynomial.

To circumvent this burden, Robust Optimization (RO) removes the necessity of knowing the parameters' probability distribution but relies on the weaker assumption that one only knows a subset of the support of the probability density function. In other words, RO only assumes to know a subset of outcomes (or scenarios) for which the density function is positive. Let $\Xi \quad \widehat{\Xi}$ be such a set and let us refer to it as the *uncertainty set*. Without any further assumption, one may see Ξ as a set of equally probable outcomes and, therefore, define a *good* decision as one which is optimal for the *worst* scenario. Thus, the RO paradigm considers the following shape for φ .

$$\varphi\ (\psi, \boldsymbol{x}) = \sup_{\boldsymbol{\xi}\,2\,\Xi} \psi(\boldsymbol{x}; \boldsymbol{\xi}) \tag{1.14}$$

Now, let $\boldsymbol{x}_{RO}$ be an optimal decision and let $\boldsymbol{\xi}$ be an outcome realizing the supremum of $\psi(\boldsymbol{x}\ ,\ )$ over Ξ, i.e., let $\boldsymbol{\xi}$ be the (/a) worst-case scenario. Then, for any scenario $\boldsymbol{\xi}^{0}\ 2\ \Xi$, we have $\psi(\boldsymbol{x}_{RO}; \boldsymbol{\xi}\ ) \quad \psi(\boldsymbol{x}_{RO}, \boldsymbol{\xi}^{0})$. Informally, were to realize in a scenario which is not the worst case, then $\boldsymbol{x}_{RO}$ would be an "even better" decision.

The RO paradigm offers many advantages over the stochastic and chance-constrained approaches. First, it will be shown in Chapter 2, which is dedicated to the state of the art of robust optimization, that the robust counterpart of a given deterministic problem belongs to the same class of complexity, i.e., the RO counterpart of polynomially solvable problems remains polynomially solvable (under mild assumption). Second, it does not rely on any assumption regarding distributions of the parameters —which make it easily applicable to practical situations. Third, choosing appropriate uncertainty sets removes outcomes with low probability, as is desired by the CCO approach (see, for instance, the Γ-uncertainty approach from Bertsimas and Sim [2004]). Finally, it may well be the only viable approach for some critical applications.

Clearly, the RO approach also has some pitfalls among which, obviously, the lack of accountability for probabilistic knowledge on is inherently present. Another known drawback is that RO tends to lead to decisions which can be regarded as *too conservative*, i.e., decisions which are too risk averse with respect to what could be considered *rational*.

1.2.4 Distributionally robust optimization

It is clear that the robust optimization framework removes the necessity for the decision maker to have a full and perfect knowledge about the probability distribution of the uncertain parameters . In the same vein, Distributionally Robust Optimization (DRO) assumes to know, rather than "the" probability distribution, a set of possible probability distributions, gathered in what is referred to as an *ambiguity set*. Let P be such a set, one takes interest in the worst possible

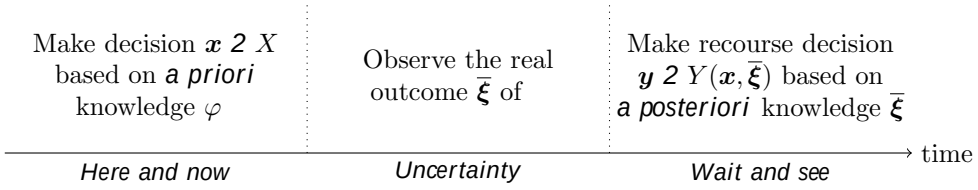


Figure 1.2: Two-stage decision flow under uncertainty

realization of the distribution with respect to the decision maker's objective while the impact of $\boldsymbol{\xi}$ is measured in terms of expectation.

More formally, let $\mathsf{E}_{\overline{\mathsf{P}}}(X)$ denote the expected value of a random variable X with respect to the probability measure $\overline{\mathsf{P}}$. Then, the DRO paradigm considers the following shape for φ .

$$\varphi\ (\boldsymbol{x}) = \sup_{\mathsf{P} \in P} \mathsf{E}_{\mathsf{P}}\left(\psi(\boldsymbol{x};\ )\right) \tag{1.15}$$

Distributionally robust optimization is a recent field motivated by some data-driven applications and can be seen, as its name suggests, as a mix between stochastic optimization and robust optimization and was heavily popularized by Delage and Ye [2010]. For a complete introduction, however, the reader may refer to the recent review Lin et al. [2022].

Remark 3 (Relation with RO)**.** *Consider an RO problem with uncertainty set Ξ, then, one can derive an equivalent optimization problem under the DRO paradigm by choosing the following ambiguity set: $P = \{\mathsf{P} : \mathsf{P}$ is a probability measure over $\Xi\}$.*

1.3 Recourse decisions

1.3.1 Two-stage decision ows

In the previous section, we considered that decision $\boldsymbol{x} \in X$ has to be taken before the realization of the unknown parameters and has to be endorsed once the latter can be observed. In many applications, decisions $\boldsymbol{x}$ are, in fact, a call for future decisions to be made, which will depend on both the *here-and-now* decision $\boldsymbol{x}$ and on the observed outcome of . This consideration leads to what is called *two-stage* models in which one accounts for the possibility of future decisions arising in a later instant while taking a decision here and now. This approach typically leads to less conservative decisions.

In Figure 1.2, we have depicted the decision flow we now consider. Here and now, decisions $\boldsymbol{x}$ has to be taken based on the *a priori* knowledge φ on . Then, once the actual outcome $\overline{\boldsymbol{\xi}}$ of is observed, so called *wait-and-see* decisions may be taken so as to react/adjust to the observed scenario. Thus, the decision maker's goal, here and now, is to decide $\boldsymbol{x}$ so that, in the future, the taken decision is regarded as *good*, taking into account that he will be able to react to the observed outcome of . This additional stage in the decision flow is referred to as the *second stage* and the associated decisions as *second-stage* decisions, *wait-and-see* decisions, or *recourse* decisions.

Clearly, the possibility to adjust one's decision after the realization of the uncertain parameters changes the cost of the considered decision, when regarded here and now. Indeed, the cost of a decision $\boldsymbol{x} \in X$, not only integrates costs which can be understood here and now, but also includes

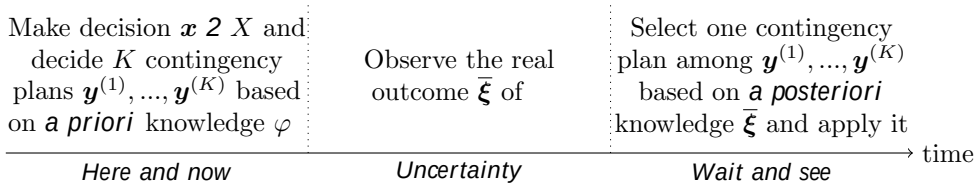


Figure 1.3: K-adaptability decision flow under uncertainty

the costs of the future decisions which are implied by x. In that sense, the set of solutions to the two-stage problem remains X. It is only the cost of each solution $x \in X$ which is changed.

More formally, let $Y \subseteq \mathbb{R}^{n_Y}$ be the set of feasible wait-and-see solutions and let $\psi^0 : X \times Y \times \widehat{\Xi} \to \overline{\mathbb{R}}$ be a function such that, for a given $x \in X$, $y \in Y$ and $\bar{\xi} \in \widehat{\Xi}$, $\psi^0(x, y; \bar{\xi})$ reflects the quality of the decisions (x, y) under the *a-posteriori*-observed scenario $\bar{\xi}$. Then a two-stage model is cast as 1.2 by defining ψ as

$$\psi(x; \bar{\xi}) = \inf_{y \in Y} \psi^0(x, y; \bar{\xi}). \tag{1.16}$$

Note that a more natural form can be obtained for cases in which constraints are also uncertain by introducing $Y(x, \bar{\xi}) = \text{dom}\left(\psi^0(x, \ ; \bar{\xi})\right)$ for $x \in X$ and $\bar{\xi} \in \widehat{\Xi}$ and to optimize $\psi^0(x, \ ; \bar{\xi})$ over $Y(x, \bar{\xi})$. In the remainder, we will refer to $Y(x, \xi)$ as the wait-and-see (second-stage, or recourse) feasible space.

For the reader's convenience, we give here the example of a general two-stage robust optimization, also called *Adjustable Robust Optimization* (ARO) in the literature

$$\inf_{x \in X} \left\{ \sup_{\xi \in \Xi} \inf_{y \in Y(x, \xi)} \psi^0(x, y; \xi) \right\}. \tag{ARO}$$

Clearly, an optimal decision x typically is such that $\varphi(x) < +\infty$, and thus, ensures that $Y(x, \xi) \neq \emptyset$ for all considered outcome ξ, i.e., there exists feasible future decisions.

1.3.2 The K-adaptability approach

As it will be examplified in the subsequent chapter dealing with solution approaches for RO, two-stage problems under uncertainty can be challenging to solve computationally, especially when the second-stage feasible space is discrete. To circumvent this fact, so-called *K-adaptability* has been introduced. It consists in deciding, here and now, decision $x \in X$ (as always) as well as K *contingency plans* $y^{(1)}, ..., y^{(K)} \in Y$ (K being an input parameter), and to select, when the uncertainty is revealed, one of the K pre-computed policies as recourse decision.

The decision flow of the K-adaptability approach is depicted in Figure 1.3. Here and now, the decision maker decides x as well as K recourse decisions based on the *a priori* knowledge φ on . When the uncertainty can be observed, the recourse decision is chosen as the best among the K contingency plans which have been selected in the first stage.

The advantage of such an approach is threefold: first, as anticipated, the resulting problem tends to be more tractable than the fully adaptable problem (i.e., the original two-stage model); second, it typically leads to more explainable solutions since the K "recourse" decisions are part of the solution; finally, it may well be the only viable solution approach for certain applications

where the contingency plans literally have to be prepared before the actual scenario is observed.

Given a two-stage uncertain optimization model $(\psi, X, \widehat{\Xi}, \varphi\ )$ with $\psi(\boldsymbol{x};\boldsymbol{\xi}) = \inf_{\boldsymbol{y}\in Y} \psi^0(\boldsymbol{x},\boldsymbol{y};\boldsymbol{\xi})$, one can easily build a K-adaptability model $(\psi^{00}, X^0, \widehat{\Xi}, \varphi\ )$ which lead to a problem cast as 1.3. This is done by fixing $X^0 = X \quad Y^K$ and ψ^{00} as follows.

$$\psi^{00}((\boldsymbol{x},\boldsymbol{y}^{(1)},...,\boldsymbol{y}^{(k)}),\overline{\boldsymbol{\xi}}) = \min_{k=1,...,K} \psi^0(\boldsymbol{x},\boldsymbol{y}^{(k)};\overline{\boldsymbol{\xi}}) \tag{1.17}$$

For simplicity, we slightly abuse notation by writing $\psi^{00}(\boldsymbol{x},\boldsymbol{Y},\overline{\boldsymbol{\xi}})$ instead of $\psi^{00}((\boldsymbol{x},\boldsymbol{y}^{(1)},...,\boldsymbol{y}^{(k)}),\overline{\boldsymbol{\xi}})$ where $\boldsymbol{Y}$ is the so-called *contingency matrix* whose columns are the contingency plans $\boldsymbol{y}^{(1)},...,\boldsymbol{y}^{(k)}$.

For the reader's convenience, we give here the example of a general K-adaptability problem in robust optimization.

$$\inf_{\substack{\boldsymbol{x}\in X \\ \boldsymbol{y}^{(1)},...,\boldsymbol{y}^{(K)}\in Y}} \sup_{\boldsymbol{\xi}\in\Xi} \min_{k=1,...,K} \psi^0(\boldsymbol{x},\boldsymbol{y}^{(k)};\boldsymbol{\xi}) \tag{K-ARO}$$

1.4 Summary

We close this chapter by discussing the different variants so far introduced regarding optimization under uncertainty. A first distinction is the choice of φ , i.e., how uncertainty is accounted for in the decision making process. Changing the definition of φ changes the paradigm, but also changes the underlying assumptions on what is known about . These paradigms are recalled in Table 1.1.

Paradigm	Risk measure, $\varphi_\Xi(\boldsymbol{x})$	Assumes knowledge of...
Deterministic	$\psi(\boldsymbol{x};\overline{\boldsymbol{\xi}})$	...the exact outcome $\overline{\boldsymbol{\xi}}$ of
Stochastic	$\mathsf{E}\,(\psi(\boldsymbol{x};\ \ ))$	...the probability distribution
Chance-constrained	$\mathsf{P}\,(P\ (\boldsymbol{x}))\,\mathsf{E}\,(\psi(\boldsymbol{x};\boldsymbol{\xi})\boldsymbol{j}P\ (\boldsymbol{x}))$ $+\delta(\boldsymbol{x}\boldsymbol{j}F_\alpha)$	...the probability distribution
Robust	$\sup_{\boldsymbol{\xi}\in\Xi} \psi(\boldsymbol{x};\boldsymbol{\xi})$	...a subset of the support of the density
Distributio. robust	$\sup_{\mathsf{P}\in P} \mathsf{E}_\mathsf{P}\,(\psi(\boldsymbol{x};\ \ ))$	...a set of probability distributions

Table 1.1: Summary of approaches for dealing with uncertainty in optimization problems

A second distinction is the way the decision flow is modeled. Formally, let $(\psi, X, \widehat{\Xi}, \varphi\ )$ be a given uncertain optimization model, we say that the model is a two-stage model when there exists (non-trivial) $Y \quad \mathsf{R}^{n_Y}$ and $\psi^0 : X \quad Y \quad \widehat{\Xi} \ \ ! \ \ \mathsf{R}$ such that $\psi(\boldsymbol{x},\boldsymbol{y};\boldsymbol{\xi}) = \inf_{\boldsymbol{y}\in Y} \psi^0(\boldsymbol{x},\boldsymbol{y};\boldsymbol{\xi})$. The model is said to be single stage otherwise. Finally, we also have introduced the idea of K-adaptability which considers the model $(\psi^{00}, X \quad Y^K, \widehat{\Xi}, \varphi\ )$ with $\psi^{00}(\boldsymbol{x},\boldsymbol{Y};\boldsymbol{\xi}) = \min_{k=1,...,K} \psi^0(\boldsymbol{x},\boldsymbol{y}^{(k)};\boldsymbol{\xi})$.

CHAPTER 2

Robust optimization

As anticipated in Chapter 1, the Robust Optimization (RO) paradigm discards the necessity of knowing the exact probability distribution governing the unknown parameters of an optimization problem. Instead, it is assumed that a subset of the support of the density function is known. This subset is referred to as the *uncertainty set* and gathers, in essence, a set of possible outcomes for the parameters which are regarded as equally probable. The decisions are then searched among those which are feasible for every scenario and the objective function is computed in the least advantageous case. For this reason, robust optimization is also referred to as worst-case optimization.

In this chapter, we will go through the current state of the art of robust optimization. First, the robust counterpart of a (parametrized) optimization problem will be introduced for both single- and two-stage problems. Then, classical uncertainty sets will be presented alongside to their conjugate support functions, which will be shown to play a crucial role in the derivation of efficient algorithmic solution methods. Subsequently, some state-of-the-art solution approaches for two-stage problems will be presented. Finally, we will cover algorithms for tackling K-adapatability problems in robust optimization.

2.1 Problem formulation

2.1.1 Single stage models

Our discussion starts with the following optimization problem, parameterized by an n_Ξ-dimensional real vector $\overline{\boldsymbol{\xi}}$ —realization of a random variable .

$$\inf f_0(\boldsymbol{x}; \overline{\boldsymbol{\xi}}) \tag{2.1}$$

$$\text{s.t. } f_i(\boldsymbol{x}; \overline{\boldsymbol{\xi}}) \quad 0 \qquad i = 1, ..., m_X \tag{2.2}$$

$$\boldsymbol{x} \; 2 \; X \quad \mathsf{R}^{n_X} \tag{2.3}$$

Note that, contrary to what was considered in Chapter 1, both the objective function (2.1) and (part of) the constraints (2.2) depend on the uncertain parameters. We refer to Remark 1 for an elaboration on the equivalence between objective uncertainty and constraint uncertainty inside the framework of Chapter 1. Simply put, one can cast problem (2.1)-(2.3) as (1.3) by choosing $\psi(\boldsymbol{x};\boldsymbol{\xi}) = f_0(\boldsymbol{x};\boldsymbol{\xi}) + \sum_{i=1}^{m_X} \delta(\boldsymbol{x} \mid \boldsymbol{x}^0 : f_i(\boldsymbol{x}^0;\boldsymbol{\xi}) \quad 0\boldsymbol{g})$. However, expressions (2.1)-(2.3) are a more natural way to express our model.

Now, assuming that the parameters is not known to exactly take value $\overline{\boldsymbol{\xi}}$ but, rather, that only a subset of the support of the density function of , noted Ξ, is known. The robust counterpart of (2.1)-(2.3) is given as follows.

$$\inf \sup_{\boldsymbol{\xi} \in \Xi} f_0(\boldsymbol{x};\boldsymbol{\xi}) \tag{2.4}$$

$$\text{s.t. } \sup_{\boldsymbol{\xi} \in \Xi} f_i(\boldsymbol{x};\boldsymbol{\xi}) \quad 0 \qquad i = 1,...,m_X \tag{2.5}$$

$$\boldsymbol{x} \in X \tag{2.6}$$

Under "natural" assumptions on functions $f_0,...,f_{m_X}$ and on the uncertainty set Ξ, it has been shown by Ben-Tal et al. [2014] that problem (2.4)-(2.6) can be reformulated as a monolithic optimization problem. We now introduce these assumptions.

Assumption A. *The uncertainty set is assumed to be a convex, nonempty, compact set.*

Assumption B. *For all $\boldsymbol{x} \in X$, functions $f_0(\boldsymbol{x};\ ),...,f_{m_X}(\boldsymbol{x};\ )$ are assumed to be concave.*

Assumption A reduces the scope of admissible uncertainty sets by requiring, among others, convexity. This assumption is justified in the context of convex optimization theory and includes polyhedral uncertainty sets, ellipsoidal uncertainty sets and generally convex uncertainty sets (see below). Assumption B assumes that functions $(f_i)_{i=0,...,m_X}$ are concave in the uncertain parameters. Both assumptions are necessary to prove the following theorem based on Fenchel duality. For more details on Fenchel duality, the reader may refer to Appendix A.

Theorem 1 (Ben-Tal et al. [2014])**.** *Under Assumption A and B, and if Slater's conditions hold, a vector $\boldsymbol{x} \in \mathbb{R}^{n_X}$ satisfies* (2.5) *if and only if the following conditions hold.*

$$\exists \boldsymbol{\pi}^{(i)} \in \mathbb{R}^{n_\Xi}, \quad \delta\left(\boldsymbol{\pi}^{(i)} \mid \Xi\right) \quad f_i\left(\boldsymbol{x};\boldsymbol{\pi}^{(i)}\right) \quad 0 \qquad i = 1,...,m_X \tag{2.7}$$

Here, $\delta(\ \mid \Xi)$ is the convex conjugate of the indicator function of Ξ and $f_i(\boldsymbol{x};\ )$ is the partial concave conjugate of f_i with respect to the unknown parameters.

Theorem 1 is a fundamental theorem in robust optimization. Indeed, it allows to turn model (2.4)-(2.6) into a monolithic optimization model which can be solved by standard optimization solvers such as Gurobi or Mosek. The robust counterpart of (2.1)-(2.3) is equivalently solved by the following optimization problem.

$$\inf \delta\left(\boldsymbol{\pi}^{(0)} \mid \Xi\right) \quad f_0\left(\boldsymbol{x};\boldsymbol{\pi}^{(0)}\right) \tag{2.8}$$

$$\text{s.t. } \delta\left(\boldsymbol{\pi}^{(i)} \mid \Xi\right) \quad f_i\left(\boldsymbol{x};\boldsymbol{\pi}^{(i)}\right) \quad 0 \qquad i = 1,...,m_X \tag{2.9}$$

$$\boldsymbol{\pi}^{(i)} \in \mathbb{R}^{n_\Xi} \qquad i = 1,...,m_X \tag{2.10}$$

$$\boldsymbol{x} \in X \tag{2.11}$$

Thus, if the deterministic problem (2.1)-(2.3) is a convex MINLP, say, then its robust counterpart can be solved by a convex MINLP as well.

Clearly, Theorem 1 is only useful if one is able to derive $\delta(\cdot|\Xi)$ and $f(\boldsymbol{x};\cdot)$ in analytical form (for all $\boldsymbol{x} \in X$). We give here an example where Ξ is a polyhedron and $(f_i)_{i=0,\dots,m_X}$ are linear functions of the uncertain parameters. For more examples, we refer to Ben-Tal et al. [2014] which details this extensively.

Example 2 (Linear setting)**.** *We consider the case where* $f_i(\boldsymbol{x};\boldsymbol{\xi}) = \boldsymbol{\xi}^T\boldsymbol{A}^i\boldsymbol{x} + b_i$ *and* $\Xi = \{\boldsymbol{\xi} \in \mathbb{R}^{n_\Xi}_+ : \boldsymbol{F}\boldsymbol{\xi} \leq \boldsymbol{g}\}$*. In this setting, one obtains the following results.*

$$\delta(\boldsymbol{\pi}^{(i)}|\Xi) = \max_{\boldsymbol{v}\in\mathbb{R}^{n_\Xi}} \left\{ {\boldsymbol{\pi}^{(i)}}^T\boldsymbol{v} \;\middle|\; \delta(\boldsymbol{v}|\Xi) \right\} = \begin{array}{lll} \max & {\boldsymbol{\pi}^{(i)}}^T\boldsymbol{v} & \\ \text{s.t.} & \boldsymbol{F}\boldsymbol{v} \leq \boldsymbol{g} & \\ & \boldsymbol{v} \geq 0 & \end{array} = \begin{array}{lll} \min & \boldsymbol{g}^T\boldsymbol{\lambda} & \\ \text{s.t.} & \boldsymbol{F}^T\boldsymbol{\lambda} \geq \boldsymbol{\pi}^{(i)} & \\ & \boldsymbol{\lambda} \geq 0 & \end{array} \tag{2.12}$$

$$f_i(\boldsymbol{x};\boldsymbol{\pi}^{(i)}) = \min_{\boldsymbol{v}\in\mathbb{R}^{n_\Xi}} \left\{ {\boldsymbol{\pi}^{(i)}}^T\boldsymbol{v} \;\middle|\; \boldsymbol{v}^T\boldsymbol{A}^i\boldsymbol{x} + b_i \right\} = b_i + \min_{\boldsymbol{v}\in\mathbb{R}^{n_\Xi}} \left\{ (\boldsymbol{\pi}^{(i)} - \boldsymbol{A}^i\boldsymbol{x})^T\boldsymbol{v} \right\} \tag{2.13}$$

$$= \begin{cases} b_i & \text{if } \boldsymbol{\pi}^{(i)} = \boldsymbol{A}^i\boldsymbol{x} \\ +\infty & \text{otherwise} \end{cases} \tag{2.14}$$

Equalities (2.12) *follow from LP duality theory (note that strong duality holds since* Ξ *is assumed to be nonempty) while* (2.13)-(2.14) *follow by inspection. Thus, by Theorem 1, we have that* (2.5) *are equivalent to the following condition.*

$$\begin{bmatrix} \min & \boldsymbol{g}^T\boldsymbol{\lambda} & \\ \text{s.t.} & \boldsymbol{F}^T\boldsymbol{\lambda} - \boldsymbol{\pi}^{(i)} & \\ & \boldsymbol{\lambda} \geq \boldsymbol{0} & \end{bmatrix} + \begin{bmatrix} \begin{cases} b_i & \text{if } \boldsymbol{\pi}^{(i)} = \boldsymbol{A}^i\boldsymbol{x} \\ +\infty & \text{otherwise} \end{cases} \end{bmatrix} \geq 0 \qquad i = 1,\dots,m_X \tag{2.15}$$

Again, by inspection, we obtain the following equivalent form.

$$\begin{bmatrix} \min & \boldsymbol{g}^T\boldsymbol{\lambda} & \\ \text{s.t.} & \boldsymbol{F}^T\boldsymbol{\lambda} - \boldsymbol{A}^i\boldsymbol{x} & \\ & \boldsymbol{\lambda} \geq \boldsymbol{0} & \end{bmatrix} \geq b_i \qquad i = 1,\dots,m_X \tag{2.16}$$

Finally, by a characterisation of minima, we have that $\boldsymbol{x}$ *is "robust feasible", i.e.,* $\boldsymbol{x}$ *satisfies* $\min\{\boldsymbol{g}^T\boldsymbol{\lambda} : \boldsymbol{F}^T\boldsymbol{\lambda} - \boldsymbol{A}^i\boldsymbol{x}, \boldsymbol{\lambda} \geq \boldsymbol{0}\} \geq b_i$*, if there exists* $\boldsymbol{\lambda}^{(i)} \geq \boldsymbol{0}$ *such that* $\boldsymbol{F}^T\boldsymbol{\lambda}^{(i)} - \boldsymbol{A}^i\boldsymbol{x}$ *and* $\boldsymbol{g}^T\boldsymbol{\lambda}^{(i)} \leq b_i$*. All in all, the robust counterpart of* (2.1)-(2.3)*, is equivalent to the following problem.*

$$\min \boldsymbol{g}^T\boldsymbol{\lambda}^{(0)} \geq b_0 \tag{2.17}$$

$$\text{s.t. } \boldsymbol{g}^T\boldsymbol{\lambda}^{(i)} \leq b_i \qquad i = 1,\dots,m_X \tag{2.18}$$

$$\boldsymbol{F}^T\boldsymbol{\lambda}^{(i)} - \boldsymbol{A}^i\boldsymbol{x} \qquad i = 0,\dots,m_X \tag{2.19}$$

$$\boldsymbol{\lambda}^{(i)} \geq \boldsymbol{0} \qquad i = 0,\dots,m_X \tag{2.20}$$

$$\boldsymbol{x} \in X \tag{2.21}$$

Note that if X is MILP-representable, then this problem is an MILP as well.

2.1.2 Two-stage models

We now turn our attention to problems which feature a two-stage decision flow. The reader may refer to Chapter 1 for an introduction to two-stage problems in decision making under uncertainty. Here and now, a set of decisions $\boldsymbol{x} \in X$ must be taken. Then, once the parameters can be observed, a second set of decisions $\boldsymbol{y} \in Y \subseteq \mathbb{R}^{n_Y}$ can be taken which depend on the first-stage decisions and on the observed parameters. Thus, let $\boldsymbol{x} \in X$ be a given first-stage decision and let $\overline{\boldsymbol{\xi}} \in \Xi$ be an observed outcome for , we introduce the set of feasible second-stage decisions, noted $Y(\boldsymbol{x}, \overline{\boldsymbol{\xi}})$, defined as follows, where $g_1, ..., g_{m_Y}$ are given functions and $Y \subseteq \mathbb{R}^{m_Y}$ is a given set specifying the domain of the $\boldsymbol{y}$-variables.

$$Y(\boldsymbol{x}, \overline{\boldsymbol{\xi}}) = \{\boldsymbol{y} \in Y : g_i(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\xi}) \leq 0 \quad i = 1, ..., m_Y\} \tag{2.22}$$

Now, based on the previous section, any uncertainty regarding the first-stage decisions can be turned into classical constraints and embedded inside X. Thus, without loss of generality, we will assume that the first stage is not subject to uncertainty. We now state the natural formulation for two-stage robust problems.

$$\inf_{\boldsymbol{x} \in X} \left\{ \sup_{\boldsymbol{\xi} \in \Xi} \inf_{\boldsymbol{y} \in Y(\boldsymbol{x}, \boldsymbol{\xi})} g_0(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\xi}) \right\} \tag{2.23}$$

In Section 2.2, we discuss several algorithmic solution approaches for this class of problems.

2.1.3 Classical uncertainty sets

In this section, we go through classical choices for uncertainty sets in robust optimization. For a more complete discussion however, we refer to the work of Li et al. [2011]. We focus here on convex uncertainty sets in line with Assumption A. In Chapter 5, we will dive into discrete uncertainty sets and introduce new solution methods for this case.

For simplicity and homogeneity, we assume that Ξ is expressed as $\Xi = \left\{\boldsymbol{\xi} \in \mathbb{R}^{n_\Xi} : \overline{\boldsymbol{\xi}} + \boldsymbol{P}\boldsymbol{\xi}, \boldsymbol{\xi} \in Z\right\}$ where $\overline{\boldsymbol{\xi}}$ is a so-called nominal value and Z is a convex compact set governing the deviations of from $\overline{\boldsymbol{\xi}}$ and $\boldsymbol{P}$ is a given matrix. This allows us to introduce generic uncertainty sets, corresponding to Z, which can be regarded as *primitive* to build more complex uncertainty sets. In what follows, n will denote the dimension of the primitive set Z, i.e., $Z \subseteq \mathbb{R}^n$ and $\boldsymbol{P} \in \mathbb{R}^{n_\Xi \quad n}$. In Table 2.1, we reported the convex conjugate of the support of each uncertainty set.

Box uncertainty

The simplest uncertainty set one can imagine is the so-called *box uncertainty set*. It is expressed using the ℓ_1-norm and an extra parameter, noted Ψ, as follows.

$$Z_1(\Psi) = \{\boldsymbol{z} \in \mathbb{R}^n : \|\boldsymbol{z}\|_1 \quad \Psi\} \tag{2.24}$$

Assuming that the random parameters are bounded, letting $\hat{\boldsymbol{\xi}}$ be the maximum deviation from $\overline{\boldsymbol{\xi}}$ so that it is known that $\boldsymbol{\xi} \in [\overline{\boldsymbol{\xi}} - \hat{\boldsymbol{\xi}}, \overline{\boldsymbol{\xi}} + \hat{\boldsymbol{\xi}}]$, then one can use Z_1 to model the *interval uncertainty set* by fixing $\Psi = 1$ and assuming that the outcome of $\boldsymbol{\xi}$ is $\overline{\boldsymbol{\xi}} + \text{diag}\left(\hat{\boldsymbol{\xi}}\right)\boldsymbol{z}$.

Polyhedral uncertainty and Γ-uncertainty

Another norm-based uncertainty set is the so-called *polyhedral uncertainty* which can be introduced using the ℓ_1-norm and a parameter Γ. It is mathematically described as follows.

$$Z_1(\Gamma) = \{\boldsymbol{z} \in \mathbb{R}^n : \|\boldsymbol{z}\|_1 \leq \Gamma\} \tag{2.25}$$

Again, for bounded unknown parameters, one can use Z_1 and Z_1 to control the sum of the deviations of $\boldsymbol{\xi}$ from its nominal value $\overline{\boldsymbol{\xi}}$. This is done through the very well-known *Γ-uncertainty set* introduced by Bertsimas and Sim [2004]. Formally, it is expressed as follows.

$$Z^0(\Gamma) = Z_1(1) \cap Z_1(\Gamma) \tag{2.26}$$

Remark 4. *We here enlight that, as originally introduced by Bertsimas and Sim [2004], the Γ-uncertainty set was defined as $Z^0(\Gamma) \cap \{0,1\}^n$. We refer to the latter as the* discrete Γ-uncertainty set *instead. We note, however, that we have* $\text{conv}\left(Z^0(\Gamma) \cap \{0,1\}^n\right) = Z^0(\Gamma)$ *so that, for uncertain MILPs which feature a single-stage decision flow, the two uncertainty sets are equivalent. In a two-stage context, however, using one set or the other is* not *equivalent. Also note that the discrete Γ-uncertainty set is, by definition, non-convex.*

Ellipsoidal uncertainty

We end this brief discussion on uncertainty sets by introducing the so-called *ellipsoidal uncertainty set* which can be used to account for correlations between the uncertain parameters. Mathematically, it is given as follows.

$$Z_2(\Omega) = \{\boldsymbol{z} \in \mathbb{R}^n : \|\boldsymbol{z}\|_2 \leq \Omega\} \tag{2.27}$$

In the following remark, we enlight an interesting bridge between chance-constrained optimization and robust optimization when Z_2 is being used.

Remark 5 (Relation with CCO)**.** *Consider the following robust constraint where $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are given vectors and matrices of agreeable size.*

$$\sup_{\boldsymbol{z} \in Z_2(\Psi)} (\boldsymbol{\mu} + \boldsymbol{\Sigma}^{1/2}\boldsymbol{z})^T\boldsymbol{x} \leq b \quad (\star) \quad \boldsymbol{\mu}^T\boldsymbol{x} + \Psi \left\|\boldsymbol{\Sigma}^{1/2}\boldsymbol{x}\right\|_2 \leq b \tag{2.28}$$

Note that the right part of the equivalence is derived thanks to the techniques introduced in Section 2.1.1. We now show that constraint (2.28) *has a direct relation with the following CCO constraint under suitable assumptions regarding $\boldsymbol{\xi}$ and Ψ. Consider the following chance constraint.*

$$\mathbb{P}\left(\boldsymbol{\xi}^T\boldsymbol{x} \leq b\right) \geq 1 - \alpha \tag{2.29}$$

Now, assume that $\boldsymbol{\xi}$ follows a normal distribution of mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\Sigma}$, i.e., $\boldsymbol{\xi} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$.

In this setting, we have that $^T\boldsymbol{x} \quad b \quad N(\boldsymbol{\mu}^T\boldsymbol{x} \quad b, \boldsymbol{x}^T\boldsymbol{\Sigma}\boldsymbol{x})$ *and, by noting* Φ *the cumulative distribution function of the normal law and* $\Phi \ ^1$ *its inverse, constraint* (2.29) *is equivalent to the following constraint.*

$$\boldsymbol{\mu}^T\boldsymbol{x} + \Phi \ ^1(1 \quad \alpha) \left\| \boldsymbol{\Sigma}^{1/2}\boldsymbol{x} \right\|_2 \quad b \tag{2.30}$$

In other words, assuming normality of the random variables and choosing $\Psi = \Phi \ ^1(1 \quad \alpha)$*, then RO and CCO are equivalent.*

Uncertainty set Z	δ $(\boldsymbol{\pi} j Z)$
$Z_1(\Psi)$	$\Psi jj\boldsymbol{\pi}jj_1$
$Z_1(\Gamma)$	$\Gamma jj\boldsymbol{\pi}jj_1$
$Z_1(1) \setminus Z_1(\Gamma)$	$\min_{\boldsymbol{w}2\mathbb{R}^n} f\Gamma jj\boldsymbol{\pi} \quad \boldsymbol{w}jj_1 + jj\boldsymbol{w}jj_1 g$
$Z_2(\Omega)$	$\Omega jj\boldsymbol{\pi}jj_2$

Table 2.1: Convex conjugate of the support functions of each "primitive" uncertainty set

2.2 Solution approaches

In this section, we present the state of the art for solving two-stage robust optimization problems. In the second part of this thesis, new solution techniques will be introduced for dealing, in particular, with integer recourse decisions or convex constraints in the second stage.

2.2.1 Linear second stage

This subsection treats the case where the second-stage feasible space is expressed as a linear optimization problem. We make this formal by introducing the following assumption.

Assumption C (Linear second stage)**.** *For a given* $\boldsymbol{x}$ *2* X *and an observed outcome* $\boldsymbol{\xi}$ *2* Ξ*, we assume that the second-stage feasible space* $Y(\boldsymbol{x}, \boldsymbol{\xi})$ *is given as follows.*

$$Y(\boldsymbol{x}, \boldsymbol{\xi}) = f\boldsymbol{y} \ 2 \ Y : \boldsymbol{T}\boldsymbol{x} + \boldsymbol{W}\boldsymbol{y} \quad \boldsymbol{h} \quad \boldsymbol{H}\boldsymbol{\xi} g \tag{2.31}$$

In other words, we assume that $g_i(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\xi}) = h_i \quad \boldsymbol{h}_{(i)}\boldsymbol{\xi} \quad \boldsymbol{w}_{(i)}\boldsymbol{y} \quad \boldsymbol{t}_{(i)}\boldsymbol{x}$ *with* $i = 1, ..., m_Y$*. Moreover, we assume that* $Y = \mathbb{R}^{n_Y}_+$ *and that* $g_0(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\xi}) = \boldsymbol{c}^T\boldsymbol{x} + \boldsymbol{d}^T\boldsymbol{y}$*.*

For simplicity of exposure, we additionally make the following assumption, referred to as the *complete recourse assumption*.

Assumption D (Complete recourse)**.** *For all* $\boldsymbol{x}$ *2* X *and all* $\boldsymbol{\xi}$ *2* Ξ*, it holds* $Y(\boldsymbol{x}, \boldsymbol{\xi})$ 6= ;*.*

Assuming that X is an MILP representable set (or more generally, any compact set), problem (2.23) can be formulated as follows.

$$\min_{\boldsymbol{x}2X} \left\{ \boldsymbol{c}^T\boldsymbol{x} + \max_{\boldsymbol{\xi}2\Xi} \min_{\boldsymbol{y}2Y(\boldsymbol{x},\boldsymbol{\xi})} \boldsymbol{d}^T\boldsymbol{y} \right\} \tag{2.32}$$

Column-and-constraint generation method

The first solution method was introduced in Zeng and Zhao [2013] and consists in the dynamic generation of columns and constraints. First, notice that the value function of the second-stage problem, i.e., $\boldsymbol{\xi} \mapsto \min_{\boldsymbol{y} \in Y(\boldsymbol{x},\boldsymbol{\xi})} \boldsymbol{d}^T\boldsymbol{y}$, is a convex function of the uncertain parameters. Thus, the following equality holds.

$$\forall \boldsymbol{x} \in X, \qquad \max_{\boldsymbol{\xi} \in \Xi} \min_{\boldsymbol{y} \in Y(\boldsymbol{x},\boldsymbol{\xi})} \boldsymbol{d}^T\boldsymbol{y} = \max_{\boldsymbol{\xi} \in \mathrm{vert}(\Xi)} \min_{\boldsymbol{y} \in Y(\boldsymbol{x},\boldsymbol{\xi})} \boldsymbol{d}^T\boldsymbol{y} \tag{2.33}$$

Here, $\mathrm{vert}\,(\Xi)$ denotes the set of extreme points of Ξ which we write as $\mathrm{vert}\,(\Xi) = \{\boldsymbol{\xi}^1, ..., \boldsymbol{\xi}^R\}$.

Then, observe that the following model is a reformulation of problem (2.23).

$$\min \boldsymbol{c}^T\boldsymbol{x} + \theta \tag{2.34}$$

$$\text{s.t. } \boldsymbol{x} \in X \tag{2.35}$$

$$\theta \geq \boldsymbol{d}^T\boldsymbol{y}^r \qquad r = 1, ..., R \tag{2.36}$$

$$\boldsymbol{y}^r \in Y(\boldsymbol{x}, \boldsymbol{\xi}^r) \qquad r = 1, ..., R \tag{2.37}$$

Indeed, for each scenario $\boldsymbol{\xi}^r$ $(r = 1, ..., R)$, a corresponding recourse decision $\boldsymbol{y}^r$ has been added to the model in such a way that the whole set of scenarios are covered by at least one recourse decision. Now, because the number of extreme points of Ξ may be prohibitively large to consider them all inside model (2.34)-(2.37), the column-and-constraint generation approach consists in solving a restricted problem where only a subset of $\mathrm{vert}\,(\Xi)$ is considered and to generate a new variable $\boldsymbol{y}^{r'}$ (with its corresponding constraints of type (2.36)-(2.37)) by solving a given separation problem which tries to identify an extreme point $\boldsymbol{\xi}^{r'}$ which is not yet covered by the wait-and-see decisions $\boldsymbol{y}^1, ..., \boldsymbol{y}^{r'-1}$.

Let $\boldsymbol{x}^{r'-1}$ denote an optimal solution of a restricted problem where only $r' - 1$ extreme points of Ξ are considered. Then the following separation problem can be used to identify an extreme point $\boldsymbol{\xi}^{r'}$ to be added to the restricted problem in order to decrease the optimality gap.

$$\max_{\boldsymbol{\xi} \in \Xi} \min_{\boldsymbol{y} \in Y(\boldsymbol{x}^{r'-1},\boldsymbol{\xi})} \boldsymbol{d}^T\boldsymbol{y} \tag{2.38}$$

Letting v denote the optimal objective value of this separation problem, a non-covered scenario is found if and only if $\theta^{r'-1} < v$.

We enlight that this method was originally presented in the linear setting, allowing for (2.33) to be established. The extension of this method to more general cases, such as convex second-stage constraints, is not straightforward since (2.33) would not necessarily hold and solution methods for (2.38) may not be at hand. This case is discussed in the last chapter of this thesis (see Chapter 7). In the technical report Zhao and Zeng [2012], this method has been extended to deal with mixed-integer second stage provided that the value function of the second stage is quasi-convex with respect to the unknown parameters. To the best of our knowledge, it has not been published in any scientific paper after 10 years.

Benders decomposition

Another broadly used approach for dealing with (2.23) is a duality-based solution technique akin to Benders decomposition. First, observe that we assumed that the second-stage feasible space is non-empty for every $\boldsymbol{x} \in X$ and every $\boldsymbol{\xi} \in \Xi$ (Assumption D). Thus, by strong LP duality, the second-stage problem has the same objective value as its dual, given as follows.

$$\left[\begin{array}{lll} \min & \boldsymbol{d}^T\boldsymbol{y} & \\ \text{s.t.} & \boldsymbol{T}\boldsymbol{x}+\boldsymbol{W}\boldsymbol{y} \quad \boldsymbol{h} \quad \boldsymbol{H}\boldsymbol{\xi} \\ & \boldsymbol{y} \in \mathbb{R}^{n_Y}_+ & \end{array}\right] = \left[\begin{array}{lll} \max & (\boldsymbol{h} \quad \boldsymbol{H}\boldsymbol{\xi} \quad \boldsymbol{T}\boldsymbol{x})^T\boldsymbol{\pi} \\ \text{s.t.} & \boldsymbol{W}^T\boldsymbol{\pi} \quad \boldsymbol{d} & \\ & \boldsymbol{\pi} \in \mathbb{R}^{m_Y}_+ & \end{array}\right] \tag{2.39}$$

Note that the dual feasible space is independent of the first-stage variables and unknown parameters. Let us denote it by D, i.e., $D = \left\{\boldsymbol{\pi} \in \mathbb{R}^{m_Y}_+ : \boldsymbol{W}^T\boldsymbol{\pi} \quad \boldsymbol{d}\right\}$. Then, the following problem is equivalent to 2.23.

$$\min_{\boldsymbol{x}\in X}\left\{\boldsymbol{c}^T\boldsymbol{x} + \max_{\boldsymbol{\xi}\in\Xi,\boldsymbol{\pi}\in D}(\boldsymbol{h} \quad \boldsymbol{H}\boldsymbol{\xi} \quad \boldsymbol{T}\boldsymbol{x})^T\boldsymbol{\pi}\right\} \tag{2.40}$$

Now, observe that $\boldsymbol{x} \mapsto \max_{\boldsymbol{\xi}\in\Xi,\boldsymbol{\pi}\in D}(\boldsymbol{h} \quad \boldsymbol{H}\boldsymbol{\xi} \quad \boldsymbol{T}\boldsymbol{x})^T\boldsymbol{\pi}$ is a convex function (maximum of convex (affine) functions). Thus, it can be approximated by its supporting hyperplanes (in fact, supporting hyperplanes of its epigraph), which, since the function is piecewise linear, are simply given as $\boldsymbol{x} \mapsto (\boldsymbol{h} \quad \boldsymbol{H}\hat{\boldsymbol{\xi}} \quad \boldsymbol{T}\boldsymbol{x})^T\hat{\boldsymbol{\pi}}$ with $(\hat{\boldsymbol{\xi}}, \hat{\boldsymbol{\pi}}) \in \Xi \quad D$. Thus, one obtains the following model for (2.23).

$$\min \boldsymbol{c}^T\boldsymbol{x} + \theta \tag{2.41}$$

$$\text{s.t. } \boldsymbol{x} \in X \tag{2.42}$$

$$\theta \quad (\boldsymbol{h} \quad \boldsymbol{H}\hat{\boldsymbol{\xi}} \quad \boldsymbol{T}\boldsymbol{x})^T\hat{\boldsymbol{\pi}} \qquad (\hat{\boldsymbol{\xi}}, \hat{\boldsymbol{\pi}}) \in (\Xi, D) \tag{2.43}$$

Now, because the number of cuts of type (2.43) may be large in practice, or hard to enumerate, dynamic row generation has been proposed. Here again, a restricted problem is considered where only a subset of constraints of type (2.43) (Benders optimality cuts) are present. Then, letting $\boldsymbol{x}$ denote an optimal solution of the restricted problem, one can identify potential violated constraints by solving the following bi-linear problem.

$$\max_{\boldsymbol{\xi}\in\Xi,\boldsymbol{\pi}\in D}(\boldsymbol{h} \quad \boldsymbol{H}\boldsymbol{\xi} \quad \boldsymbol{T}\boldsymbol{x} \ )^T\boldsymbol{\pi} \tag{2.44}$$

Note that, in this case, Assumption D can be relaxed to consider the more general case where the second-stage problem is not always feasible. This gives rise to so-called *Benders feasibility cuts* which rule out first-stage decisions leading to infeasible second-stage decisions.

A ne Decision Rules (safe approximation)

While the first two solution approaches we presented are exact methods, we now turn our attention to a very popular so-called *safe approximation* of two-stage robust problems. The starting point is the equivalence between problem (2.23) and the following one.

$$\min \boldsymbol{c}^T\boldsymbol{x} + \max_{\boldsymbol{\xi}\in\Xi} \boldsymbol{d}^T\boldsymbol{y}(\boldsymbol{\xi}) \tag{2.45}$$

$$\text{s.t. } \boldsymbol{x} \in X \tag{2.46}$$

$$\boldsymbol{y} : \Xi \; ! \; Y \tag{2.47}$$

$$\boldsymbol{y}(\boldsymbol{\xi}) \; 2 \; Y(\boldsymbol{x}, \boldsymbol{\xi}) \quad 8\boldsymbol{\xi} \; 2 \; \Xi \tag{2.48}$$

Here, the optimization is done over X and over the set of all functions from Ξ to Y. Informally speaking, function $\boldsymbol{y}(\ )$ now returns the decided recourse decision to use in the scenario corresponding to its argument. Thus, we must enforce that, for all $\boldsymbol{\xi} \; 2 \; \Xi$, it holds $\boldsymbol{y}(\boldsymbol{\xi}) \; 2 \; Y(\boldsymbol{x}, \boldsymbol{\xi})$. In the literature, function $\boldsymbol{y}(\ )$ is called a Decision Rule (DR).

Unfortunately, optimizing over a set of functions is challenging. Instead, one may restrict its attention to cases in which $\boldsymbol{y}(\ )$ is an affine function of the unknown parameters, thus obtaining a (math-)heuristic approach. This leads to the Affine Decision Rule (ADR) approximation. This approximation is obtained by the following single-stage robust optimization problem.

$$\min \boldsymbol{c}^T \boldsymbol{x} + \max_{\boldsymbol{\xi} 2 \Xi} \boldsymbol{d}^T (\overline{\boldsymbol{y}} + \boldsymbol{Y}\boldsymbol{\xi}) \tag{2.49}$$

$$\text{s.t. } \boldsymbol{x} \; 2 \; X, \overline{\boldsymbol{y}} \; 2 \; \mathsf{R}^{n_Y}, \boldsymbol{Y} \; 2 \; \mathsf{R}^{n_Y \quad n_\Xi} \tag{2.50}$$

$$\boldsymbol{T}\boldsymbol{x} + \boldsymbol{W}(\overline{\boldsymbol{y}} + \boldsymbol{Y}\boldsymbol{\xi}) \quad \boldsymbol{h} \quad \boldsymbol{H}\boldsymbol{\xi} \qquad 8\boldsymbol{\xi} \; 2 \; \Xi \tag{2.51}$$

Here, $\boldsymbol{x}, \overline{\boldsymbol{y}}$ and $\boldsymbol{Y}$ are the decision variables and $\boldsymbol{y}(\boldsymbol{\xi}) = \overline{\boldsymbol{y}} + \boldsymbol{Y}\boldsymbol{\xi}$. By using the classical techniques introduced in Section 2.1.1, a monolithic optimization problem can be derived and directly solved using standard optimization algorithms. For a discussion on optimality conditions for ADR, we refer to Marandi and den Hertog [2017]. Unfortunately, in most cases, the ADR approach only offers a safe approximation of (2.23) and fails to return an optimal solution.

We enlight that this approach cannot be extended to cases where the second stage is defined by means of (nonlinear) convex functions. Indeed, this would lead to formulations in which Assumption B fails to be fulfilled. Moreover, generalizing it to mixed-integer second stages is also impossible as it would require imposing integrality of $\overline{\boldsymbol{y}} + \boldsymbol{Y}\boldsymbol{\xi}$ for all $\boldsymbol{\xi} \; 2 \; \Xi$.

2.2.2 Convex second stage

Regarding the convex case, occurring when $Y(\boldsymbol{x}, \boldsymbol{\xi})$ is defined by convex functions of the first- and second-stage variables, the techniques introduced for the linear case are not directly applicable, and the scientific literature is much more sparse.

In Boni and Ben-Tal [2008], the authors consider two-stage problems with ellipsoidal uncertainty and conic quadratic second-stage constraints. They show that an optimal ADR can be obtained by means of a semidefinite programing problem. Unfortunately, this approach cannot be generalized to other uncertainty sets and other convex functions as it relies on the strong duality result of a non-convex problem which, as is known to all, does not hold in the general case.

Then, Takeda et al. [2007] considers two-stage problems where the uncertainty set is expressed as the convex hull of a finite set of points and report conditions under which their problem can be reduced to a single-stage problem. However, the size of the resulting problem depends on the number of points defining the uncertainty set which, in general, is exponentially large.

In de Ruiter et al. [2022], the authors derive a dualized problem for a class of convex two-stage problems by extending an approach from Bertsimas and de Ruiter [2016]. This dualized problem being linear in the uncertain parameters, they use ADR techniques to obtain a tight approximation

for the objective value. Moreover, they show that a primal *feasible* ADR can be derived from an optimal dual ADR. This is in contrast with the results of Bertsimas and de Ruiter [2016] where an *optimal* primal ADR could be derived from the dual.

In Chapter 7, we show that the column-and-constraint generation algorithm introduced for the linear case can be adapted to the more general convex case under some limiting assumptions on the uncertainty set which is used.

2.2.3 Mixed-integer second stage

For cases where the second stage contains integer decisions, all the previously introduced methods fail to be generalized in a straightforward manner. The scientific literature regarding mixed-integer second stage, which do gather a very large set of real-world applications, is very sparse and no real outsiding method can be pointed out.

In this section, we make the following assumption that the second-stage feasible space contains integer decisions.

Assumption E (Mixed-integer second stage)**.** *Same as Assumption C, but with* $Y \quad \mathsf{R}_+^{n_Y}$ *instead of* $Y = \mathsf{R}_+^{n_Y}$*. In particular, we allow for* $Y = f0, 1g^{n_Y}$*.*

Binary decision rules (approximation)

In Bertsimas and Georghiou [2017], the authors introduce binary decision rules which, in the vein of ADR approximation, replaces the decision rule $\boldsymbol{y}(\boldsymbol{\xi})$ with some parameterized function. In particular, they use linearly parameterized binary decision rules and derive exact reformulations for the decision rule problem. The general shape of such decision rules is as follows.

$$\boldsymbol{y}(\boldsymbol{\xi}) = \boldsymbol{Y}\boldsymbol{p}(\boldsymbol{\xi}), \quad \boldsymbol{Y} \; 2 \; \mathsf{Z}^{n_Y \quad g} \tag{2.52}$$

$$\mathbf{0} \quad \boldsymbol{y}(\boldsymbol{\xi}) \quad \boldsymbol{e} \tag{2.53}$$

where $p_i(\boldsymbol{\xi}) = \mathbb{1}(\boldsymbol{\alpha}_i^T \boldsymbol{\xi} \quad \beta_i)$ for $i = 1, ..., g$. Here, $\boldsymbol{Y}$, $\boldsymbol{A}$ and $\boldsymbol{\beta}$ are to be decided. Unfortunately, this approach fails to return optimal solutions in the general case. Moreover, solving the obtained approximation may result in the solution of an exponentially large problem implying a possible trade-off between efficiency and optimality.

Column generation (for objective uncertainty)

In the special case where only the objective function is parametrized by the unknown parameters, a decomposition-based solution approach was introduced in Arslan and Detienne [2021] under some restrictive assumptions made on the linking constraints between the first- and second-stage variables. In particular, we make the following assumption which replaces Assumption C and E.

Assumption F (Objective uncertainty)**.** *For a given* $\boldsymbol{x}$ *2* X *and an observed outcome* $\boldsymbol{\xi}$ *2* Ξ*, we assume that the second-stage feasible space* $Y(\boldsymbol{x}, \boldsymbol{\xi})$ *is independent of* $\boldsymbol{\xi}$*, i.e.,* $Y(\boldsymbol{x}, \boldsymbol{\xi}) = Y(\boldsymbol{x})$*, and given as follows.*

$$Y(\boldsymbol{x}) = f\boldsymbol{y} \; 2 \; Y : \boldsymbol{T}\boldsymbol{x} + \boldsymbol{W}\boldsymbol{y} \quad \boldsymbol{h}g \tag{2.54}$$

In other words, we assume that $g_i(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\xi}) = h_i \quad \boldsymbol{w}_{(i)}\boldsymbol{y} \quad \boldsymbol{t}_{(i)}\boldsymbol{x}$ *with* $i = 1, ..., m_Y$*. Moreover, we assume that* $Y \quad \mathsf{R}^{n_Y}$ *and that* $g_0(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\xi}) = \boldsymbol{c}^T\boldsymbol{x} + \boldsymbol{\xi}^T\boldsymbol{Q}\boldsymbol{y}$*.*

Under Assumption F, problem (2.23) can be formulated as follows.

$$\min_{\boldsymbol{x}\,2\,X} \left\{ \boldsymbol{c}^T\boldsymbol{x} + \max_{\boldsymbol{\xi}\,2\,\Xi} \min_{\boldsymbol{y}\,2\,Y(\boldsymbol{x})} \boldsymbol{\xi}^T\boldsymbol{Q}\boldsymbol{y} \right\} \tag{2.55}$$

Then, observe that the second-stage feasible space can be replaced by its convex hull, by linearity of the objective function. Indeed, when $\boldsymbol{\xi}$ is fixed, $\boldsymbol{y}$ 7! $\boldsymbol{\xi}^T\boldsymbol{Q}\boldsymbol{y}$ is a linear function. Moreover, note that conv $(Y(\boldsymbol{x}))$ is a convex compact set. Thus, one may apply the Von Neumann min-max theorem to swap the max and min operators in (2.55). Formally, this is done as follows.

$$\min_{\boldsymbol{x}\,2\,X} \left\{ \boldsymbol{c}^T\boldsymbol{x} + \max_{\boldsymbol{\xi}\,2\,\Xi} \min_{\boldsymbol{y}\,2\,\mathrm{conv}(Y(\boldsymbol{x}))} \boldsymbol{\xi}^T\boldsymbol{Q}\boldsymbol{y} \right\} = \inf_{\substack{\boldsymbol{x}\,2\,X \\ \boldsymbol{y}\,2\,\mathrm{conv}(Y(\boldsymbol{x}))}} \left\{ \boldsymbol{c}^T\boldsymbol{x} + \max_{\boldsymbol{\xi}\,2\,\Xi} \boldsymbol{\xi}^T\boldsymbol{Q}\boldsymbol{y} \right\} \tag{2.56}$$

In Arslan and Detienne [2021], the authors consider that Ξ is defined by m_Ξ linear constraints, say $\Xi = f\boldsymbol{\xi}\ 2\ \mathsf{R}^{n_\Xi}_+ : \boldsymbol{F}\boldsymbol{\xi} \quad \boldsymbol{g}g$, allowing to turn the inner maximization problem into a minimization problem by strong LP duality (note that Ξ is assumed to be non-empty). Thus, one obtains the following reformulation of (2.55).

$$\min \boldsymbol{c}^T\boldsymbol{x} + \boldsymbol{g}^T\boldsymbol{\lambda} \tag{2.57}$$

$$\text{s.t. } \boldsymbol{x}\ 2\ X \tag{2.58}$$

$$\boldsymbol{F}^T\boldsymbol{\lambda} \quad \boldsymbol{Q}\boldsymbol{y} \tag{2.59}$$

$$\boldsymbol{y}\ 2\ \mathrm{conv}\,(Y(\boldsymbol{x})) \tag{2.60}$$

$$\boldsymbol{\lambda}\ 2\ \mathsf{R}^{m_\Xi}_+ \tag{2.61}$$

A challenging issue with formulation (2.57)-(2.61) is enforcing constraint (2.60) which requires that $\boldsymbol{y}$ belongs to the convex hull of a set of points depending on some decision variables. To tackle this issue, Arslan and Detienne [2021] introduce a restrictive assumption on $Y(\ )$ so as to move out the dependency on the $\boldsymbol{x}$ variables. We now state this assumption.

Assumption G (Interdiction linking constraints)**.** *We assume that* $\boldsymbol{x}$ *can be split into* $(\boldsymbol{x}^1, \boldsymbol{x}^2)$ *such that* $\boldsymbol{x}\ 2\ X$ *implies that* $\boldsymbol{x}^1\ 2\ f0, 1g^{p_X}$*. Moreover, we assume that* $Y(\boldsymbol{x}) = Y(\boldsymbol{x}^1)$ *and* $\boldsymbol{T} = [\boldsymbol{I}; \boldsymbol{0}]$*,* $\boldsymbol{W} = [\ \ \boldsymbol{I}; \boldsymbol{W}^0]$ *and* $\boldsymbol{h} = [\boldsymbol{0}; \boldsymbol{h}^0]$ *with* $\boldsymbol{W}^0$ *and* $\boldsymbol{h}^0$ *matrices of agreeable size. In other words, we assume that* $Y(\ )$ *is expressed as follows.*

$$Y(\boldsymbol{x}) = Y(\boldsymbol{x}^1) = \left\{ \boldsymbol{y}\ 2\ Y : \boldsymbol{W}^0\boldsymbol{y} \quad \boldsymbol{h}^0 \text{ and } \boldsymbol{y}^1 \quad \boldsymbol{x}^1 \right\} \tag{2.62}$$

Here, $\boldsymbol{y}^1$ *denotes the sub-vector of* $\boldsymbol{y}$ *whose components are linked to* $\boldsymbol{x}$*.*

Then, the authors exploit the fact that, under Assumption G, the following equality holds: $\mathrm{conv}\left(Y(\boldsymbol{x}^1)\right) = \mathrm{conv}\,(f\boldsymbol{y}\ 2\ Y : \boldsymbol{W}\boldsymbol{y} \quad \boldsymbol{h}^0g) \setminus f\boldsymbol{y} : \boldsymbol{y}^1 \quad \boldsymbol{x}^1g$. For convenience, let $Y^0 = f\boldsymbol{y}\ 2\ Y : \boldsymbol{W}\boldsymbol{y} \quad \boldsymbol{h}^0g$. Thus, the following model is an exact reformulation of (2.55) under Assumption G.

$$\min \boldsymbol{c}^T\boldsymbol{x} + \boldsymbol{g}^T\boldsymbol{\lambda} \tag{2.63}$$

$$\text{s.t. } (\boldsymbol{x}^1, \boldsymbol{x}^2) \in X \tag{2.64}$$
$$(\boldsymbol{y}^1, \boldsymbol{y}^2) \in \text{conv}\,(Y^0) \tag{2.65}$$
$$\boldsymbol{y}^1 \quad \boldsymbol{x}^1 \tag{2.66}$$
$$\boldsymbol{F}^T\boldsymbol{\lambda} \quad \boldsymbol{Q}[\boldsymbol{y}^1; \boldsymbol{y}^2] \tag{2.67}$$
$$\boldsymbol{\lambda} \in \mathbb{R}_+^{n_\Xi} \tag{2.68}$$

Finally, assuming that X is defined by linear constraints, the obtained model can be solved using standard linear column generation techniques, leading to a branch-and-price algorithm.

In Chapter 3, we generalize their work to handle a more general case, removing any assumption on the linking constraints and allowing for convex second-stage feasible space. In Chapter 4, we apply the resulting framework to a two-stage scheduling problem with uncertain job failure.

Finally, in Chapter 5, we introduce a new result regarding two-stage robust problems with discrete uncertainty sets and mixed-integer second-stage decisions. We will show that, in this case, any problem with uncertain constraints can be transformed into a problem where the uncertain parameters only interfere in the objective function. We then introduce a new solution method for this hard class of problems by generalizing the work of Kämmerling and Kurtz [2020], a similar approach to that of Arslan and Detienne [2021].

2.2.4 The K-adaptability problem

In this section, we consider the K-adaptability problem for (2.23) when the second-stage feasible space is an MILP. We now turn our attention to two solution methods which have been proposed in the literature to solve (2.69). The first approach does not rely on any assumption stronger than A and C. The second one, instead, assumes that the uncertainty only interferes in the objective function.

Scenario-based Branch-and-Bound

In this subsection, we work under Assumption A and C. The K-adaptability problem can be formulated as follows.

$$\inf_{\substack{\boldsymbol{x} \in X \\ \boldsymbol{Y} \in Y^K}} \left\{ \boldsymbol{c}^T\boldsymbol{x} + \max_{\boldsymbol{\xi} \in \Xi} \min_{k=1,\dots,K} \left\{ \boldsymbol{d}^T\boldsymbol{y}^{(k)} + \delta(\boldsymbol{y}^{(k)} | Y(\boldsymbol{x}, \boldsymbol{\xi})) \right\} \right\} \tag{2.69}$$

The method, which was introduced in Subramanyam et al. [2019], relies on the following key idea: choosing K contingency plans to address all scenarios inside Ξ reduces to build a partition of Ξ of size K for which one has to decide an optimal policy for each element of the partition. Now, let $\Xi^1, \dots, \Xi^K$ be such that $\Xi^1 \cup \dots \cup \Xi^K \supseteq \Xi$ and consider the following optimization problem.

$$\inf \boldsymbol{c}^T\boldsymbol{x} + \theta \tag{2.70}$$
$$\text{s.t. } \boldsymbol{x} \in X \tag{2.71}$$
$$\boldsymbol{y}^{(k)} \in Y \qquad k = 1, \dots, K \tag{2.72}$$
$$\theta \quad \boldsymbol{d}^T\boldsymbol{y}^{(k)} \qquad k = 1, \dots, K \tag{2.73}$$
$$\boldsymbol{y}^{(k)} \in Y(\boldsymbol{x}, \boldsymbol{\xi}) \qquad \forall \boldsymbol{\xi} \in \Xi^k, k = 1, \dots, K \tag{2.74}$$

Intuitively, $\boldsymbol{y}^{(k)}$ denotes the policy to apply for all scenarios inside Ξ^k. Clearly, a crucial point is that one does not know, *a priori*, an optimal set $\{\Xi^1, ..., \Xi^K\}$ for which (2.69) and (2.70)-(2.74) are equivalent.

In Subramanyam et al. [2019], the authors suggest to start with an initial set of empty subsets, i.e., with $\Xi^1 = \emptyset, ..., \Xi^K = \emptyset$. And to dynamically generate scenarios so as to build an appropriate partition of Ξ. The obtained algorithm is a Branch-and-Bound algorithm where each parent node has K children. Now, let $(\theta, \boldsymbol{x}, \boldsymbol{Y})$ be an optimal solution to (2.70)-(2.74). The following separation identifies a scenario for which none of the contingency plans inside $\boldsymbol{Y}$ are feasible.

$$\max_{\boldsymbol{\xi} \in \Xi} \min_{k=1,...,K} \max_{i=1,...,m_Y} \{h_i - \boldsymbol{h}_{(i)}\boldsymbol{\xi} - \boldsymbol{w}_{(i)}\boldsymbol{y} - \boldsymbol{t}_{(i)}\boldsymbol{x}\} \tag{2.75}$$

If the separation problem identifies one such scenario, say $\boldsymbol{\xi}$, then K new nodes in the branch-and-bound tree are created such that, for each $k \in \{1, ..., K\}$, node k is such that $\boldsymbol{\xi} \in \Xi^k$. The method is proved to converge asymptotically to an optimal solution of (2.69).

Linearization for objective uncertainty

In this section, we assume that the uncertain parameters only interfere in the objective function. Thus, we work in the framework implied by Assumption F. Thus, the K-adaptability problem is cast as

$$\inf_{\substack{\boldsymbol{x} \in X \\ \boldsymbol{Y} \in [Y(\boldsymbol{x})]^K}} \left\{ \boldsymbol{c}^T\boldsymbol{x} + \max_{\boldsymbol{\xi} \in \Xi} \min_{k=1,...,K} \boldsymbol{\xi}^T\boldsymbol{Q}^T\boldsymbol{y}^{(k)} \right\}. \tag{2.76}$$

In this framework, Hanasusanto et al. [2015] introduced a solution method akin to what is done for the fully adaptable case of Arslan and Detienne [2021]. Their approach, however, is restricted to the case where $Y = \{0,1\}^{n_Y}$ as it will become clear later on. First, the inner minimization problem is reformulated as

$$\min_{k=1,...,K} \boldsymbol{\xi}^T\boldsymbol{Q}^T\boldsymbol{y}^{(k)} = \min_{\boldsymbol{\alpha} \in \{0,1\}^K} \sum_{k=1}^{K} \alpha_k \boldsymbol{\xi}^T\boldsymbol{Q}^T\boldsymbol{y}^{(k)}. \tag{2.77}$$

Then, by linearity of the objective function, "$\boldsymbol{\alpha} \in \{0,1\}^K$" can be replaced by "$\boldsymbol{\alpha} \in [0,1]^K$". In turn, the Von Neumann min-max theorem may again be applied to obtain the following formulation.

$$\inf \boldsymbol{c}^T\boldsymbol{x} + \max_{\boldsymbol{\xi} \in \Xi} \sum_{k=1}^{K} \alpha_k \boldsymbol{\xi}^T\boldsymbol{Q}^T\boldsymbol{y}^{(k)} \tag{2.78}$$

$$\text{s.t. } \boldsymbol{x} \in X \tag{2.79}$$

$$\boldsymbol{y}^{(k)} \in Y(\boldsymbol{x}) \qquad k = 1, ..., K \tag{2.80}$$

$$\boldsymbol{\alpha} \in [0,1]^K \tag{2.81}$$

The remaining steps are similar to that of Arslan and Detienne [2021]. Indeed, assuming that $\Xi = \{\boldsymbol{\xi} \in \mathbb{R}^{n_\Xi}_+ : \boldsymbol{F}\boldsymbol{\xi} \geq \boldsymbol{g}\}$, LP duality can be used to turn the remaining maximization into minimization. The last step in order to reformulate (2.76) as a classical optimization problem is to linearize every product between α_k and $\boldsymbol{y}^{(k)}$ for $k = 1, ..., K$. This is done using standard

linearization techniques and are the key argument for requiring $Y = \mathsf{f}0, 1\mathsf{g}^{n_Y}$. One should note that if $K \quad n_Y + 1$, Hanasusanto et al. [2015] and Arslan and Detienne [2021] are equivalent (see Carathéodory's theorem).

Part II

Contributions

CHAPTER 3

Mixed-integer problems with objective uncertainty

Boris Detienne, Henri Lefebvre, Enrico Malaguti, Michele Monaci[1]

In this chapter, we study optimization problems where some cost parameters are not known at decision time and the decision flow is modeled as a two-stage process within a robust optimization setting. We address general problems in which all constraints (including those linking the first and the second stages) are defined by convex functions and involve mixed-integer variables, thus extending the existing literature to a much wider class of problems. We show how these problems can be reformulated using Fenchel duality, allowing us to derive an enumerative exact algorithm, for which we prove asymptotic convergence in the general case, and finite convergence in the binary case.

An implementation of the resulting algorithm, embedding a column generation scheme, is then computationally evaluated on a variant of the Capacitated Facility Location Problem with unkown transportation costs, using instances that are derived from the existing literature. To the best of our knowledge, this is the first approach providing results on the practical solution of this class of problems.

3.1 Introduction

Recall from Chapter 2 that a general two-stage robust optimization problem is formulated as

$$\inf_{\boldsymbol{x} \in X} \left\{ \sup_{\boldsymbol{\xi} \in \Xi} \inf_{\boldsymbol{y} \in Y(\boldsymbol{x},\boldsymbol{\xi})} g_0(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\xi}) \right\}. \tag{3.1}$$

Here, X denotes the set of feasible first-stage decisions (or, here-and-now decisions), Ξ is a given uncertainty set (i.e., a subset of the support of the density function of the random parameters)

[1] The content of this chapter has been submitted to *EJOR - European Journal of Operations Research* and is currently under revision.

and, $Y(\boldsymbol{x}, \boldsymbol{\xi})$ denotes the set of second-stage decisions (or, wait-and-see decisions) defined for a given first-stage decision $\boldsymbol{x} \in X$ and a given scenario $\boldsymbol{\xi} \in \Xi$. We refer to Chapter 2 for an extensive state-of-the-art description regarding this general case.

An important special case of (3.1) arises when uncertainty affects the objective function only, i.e., $Y(\boldsymbol{x}, \boldsymbol{\xi}) = Y(\boldsymbol{x}), \forall \boldsymbol{\xi} \in \Xi$. For this specific case, Kämmerling and Kurtz [2020] proposed an oracle-based algorithm relying on a hull relaxation combining the first- and second-stage feasible spaces embedded within a branch-and-bound framework. However, this approach applies to purely binary variables and linear constraints only. On the other hand, Arslan and Detienne [2021] proposed an exact MILP reformulation of the problem in case of linear linking constraints that involve binary variables only. Besides solving the problem by means of a branch-and-price algorithm, a further contribution of Arslan and Detienne [2021] is proving the NP-completeness of the problem in this setting.

In the setting where uncertainty affects the objective function only, our analysis shows that further effort is needed to tackle more general cases, in particular when linking constraints are defined by nonlinear functions or involve both integer and continuous variables. Similarly, to the best of our knowledge, the case in which the objective function is nonlinear has not been considered yet. This work contributes in filling this gap, as we consider two-stage robust problems with objective uncertainty, convex constraints and mixed-integer first and second stage. By extending in a non-trivial way some recent results from the two-stage stochastic optimization literature (see Sherali and Fraticelli [2002], Sherali and Zhu [2006] and Li and Grossmann [2019]), we obtain a relaxation of the problem, and analyze its tightness for different special cases. This relaxation can be embedded within a branch-and-bound scheme thus producing an exact solution approach, for which we prove asymptotic convergence in the general case, and finite convergence in the integer case.

This chapter is organized as follows. In Section 3.2 we formally introduce the class of problems we are considering throughout this work, whereas, in Section 3.3, we present a relaxation of the problem, and an effective algorithm for its solution. We then derive sufficient conditions for the relaxation to coincide with the original problem in a mixed-integer context. So as to close the optimality gap, we introduce a branch-and-bound algorithm which embeds a spatial branching mechanism on continuous first-stage variables, and prove asymptotic convergence of the overall algorithm in presence of continuous first-stage decisions and finite ε-convergence in case of integer first-stage decisions. Finally, Section 3.4 applies the proposed method to a capacitated facility location problem with congestion.

3.2 Problem de nition

As anticipated, our goal is to solve problem (3.1) with objective uncertainty, convex constraints and mixed-integer first and second stages.

For the sake of clarity, let us first introduce several sets. Set $I = \{1, \ldots, n_X\}$ denotes the set of indices for the first-stage variables, and is partitioned into two sets I_I and I_C: variables whose index belongs to I_I are required to take integer values, while those whose index belongs to I_C are continuous variables, i.e., wlog, $X \subseteq \mathbb{R}^{|I_C|} \times \mathbb{Z}^{|I_I|}$. Similarly, we introduce set $J = \{1, \ldots, n_Y\}$ as the indices for the second-stage variables and partition this set into J_I and J_C, i.e., wlog,

Y R^{jJ_Cj} Z^{jJ_Ij}. Finally, we introduce set $U = f1, \ldots, n_\Xi g$ as the index set for the uncertain variables, i.e., Ξ R^{jUj}.

We now explicit some assumptions on the problem.

Assumption H (Objective uncertainty)**.** *For all ξ 2 Ξ and x 2* cont (X)*, $Y(\xi, x) = Y(x)$.*

Assumption I (Convexity)**.**

1. cont (X) *is compact and convex;*
2. *The uncertainty set Ξ is a nite-dimensional, bounded convex set;*
3. *For all x 2* cont (X)*,* cont $(Y(x))$ *is a nite-dimensional, bounded convex set;*
4. *The objective function g_0 is a concave function of the uncertain parameters and a convex function of the rst- and second-stage decisions, i.e., $g_0^{x,y} : \xi$ 7! $g_0(\xi, x, y)$ is a concave function for all xed x 2* cont (X) *and y 2* cont $(Y(x))$ *and $g_0^{\xi} : (x, y)$ 7! $g_0(\xi, x, y)$ is a convex function for all xed ξ 2 Ξ.*

Assumption J (Complete recourse)**.** *For every (relaxed) rst-stage decision, there exists at least one feasible second-stage decision, i.e., for every x 2* cont (X)*, $Y(x)$ is a non-empty set.*

Assumption K (Boundedness)**.**

1. *The objective function g_0 is bounded over the rst- and second-stage feasible space, i.e., for all xed ξ 2 Ξ, f$(x, y) : x$ 2* cont (X) *, y 2* cont $(Y(x))$g dom $\left(g_0^{\xi}\right)$
2. *For all $(x, y) : x$ 2* cont (X) *and y 2* cont $(Y(x))$*,* relint(Ξ) \ dom $(g_0^{x,y})$ 6= ;

Assumption L (Separability)**.** *Let $Q = f1, \ldots, qg$.*

1. *The objective function g_0 can be expressed as a sum of q functions, i.e., there exist q functions $(\phi_i : R^{jIj+jJj+jUj}$! R$)_{i2Q}$ such that $g_0(x, y; \xi) = \sum_{i2Q} \phi_i(x, y; \xi)$ for all x 2 X, y 2 $Y(x)$ and all ξ 2 Ξ.*
2. *For all i 2 Q, ϕ_i is separable in ξ and (x, y) meaning that there exists functions $(w_i : R^{jUj}$! R$)_{i2Q}$ and $(\rho_i : R^{jIj+jJj}$! R$)_{i2Q}$ such that $\phi_i(x, y; \xi) = w_i(\xi)\rho_i(x, y)$. In addition, we assume that $w_i(\ )$ is a concave function and $\rho_i(\ )$ is a convex function.*

A few remarks regarding these assumptions are necessary. First, note that Assumptions H and I are here to define what we refer to as *convex mixed-integer robust problems with objective uncertainty*. It is important to highlight that the word "convex" is here to suggest that all involved functions are convex with respect to the first- and second-stage variables. Yet, in general, even under these assumptions, problem (3.1) may fail to have a straightforward convex MINLP formulation, as function $h : x$ 7! $\max_{\xi 2 \Xi} \min_{y 2 Y(x)} g_0(x, y; \xi)$ is not necessarily a convex function over the continuous relaxation of X. We give here a small example.

Example 3 (nonconvex MINLP)**.** *Consider the following rst- and second-stage feasible regions:*

$$X = [0, 1] \text{ and } Y(x) = \left\{ y \text{ 2 } f0, 1g^2 \;\middle|\; \begin{array}{ccc} y_1 + y_2 & 1 & \\ y_1 & 1 & x \end{array} \right\}$$

By inspection, we have that $(y_1, y_2) = (0, 0)$ *and* $(y_1, y_2) = (0, 1)$ *are always feasible second-stage solutions, while* $(y_1, y_2) = (1, 0)$ *is feasible only when* $x = 0$*. Fixing the uncertainty set* $\Xi = [0, 1]$*, we take interest in the following* convex mixed-integer two-stage robust problem*:*

$$\min_{x \in [0,1]} G(x) \text{ with } G : x \mapsto \max_{\xi \in [0,1]} \min_{(y_1,y_2) \in Y(x)} \xi(\quad 2y_1 + y_2 + 1)$$

Though every involved functions are convex (in fact, a ne), the following holds:

$$G(x) = \begin{cases} \max_{\xi \in [0,1]} \min\{\xi; 2\xi; \quad \xi\} = 0 & \text{if } x = 0 \\ \max_{\xi \in [0,1]} \min\{\xi; 2\xi\} = 1 & \text{if } x > 0 \end{cases} = \mathbb{1}(x > 0)$$

Clearly, G *fails to be convex over* $[0, 1]$ *which ends our example.*

Assumption J is a standard assumption in the two-stage optimization literature, and is known to be easy to enforce as soon as the considered problem is bounded, which is implied by Assumption K.1. Assumption K.2 is not restrictive in practice, and will be used in the proof of lemma (2). Finally, Assumption L is structural to our work, and implies the following remark.

Remark 6. *The assumption that* $\rho_i(\,)$ *is a convex function (at most a ne) for all* $i \in Q$ *is without loss of generality.*

Proof. Let $i \in Q$ such that $\varphi_i(\,)$ is concave, then, to fulfill Assumption I.4, $w_i(\boldsymbol{\xi})$ must be negative for all $\boldsymbol{\xi} \in \Xi$. Thus, one may equivalently replace $w_i(\,)$ by $\quad w_i(\,)$ and $\rho_i(\,)$ by $\quad \rho_i(\,)$. □

Remark 7. *For all* $i \in Q$ *such that* $\rho_i(\,)$ *(resp.* $w_i(\,)$*) is not single-signed, then* $w_i(\,)$ *(resp.* $\rho_i(\,)$*) is a ne.*

Remark 8. *For all* $i \in Q$ *such that* $\rho_i(\,)$ *(resp.* $w_i(\,)$*) is not a ne, then* $w_i(\,)$ *(resp.* $\rho_i(\,)$*) is a non-negative function.*

For the reader's convenience, we now give some examples of functions which fulfill or violate the separability assumption (i.e., Assumption L).

Example 4 (Fulfilling Assumption L)**.** *We give here some examples of functions which satisfy Assumption L. For simplicity, we denote* $\boldsymbol{z} = (\boldsymbol{x}, \boldsymbol{y})$*.*

- *Uncertain linear functions of the form* $(\boldsymbol{\xi}, \boldsymbol{z}) \mapsto \boldsymbol{\xi}\boldsymbol{A}\boldsymbol{z}$ *where* $\boldsymbol{A}$ *is a given real matrix;*
- *Diagonal uncertain convex quadratic form* $(\boldsymbol{\xi}, \boldsymbol{z}) \mapsto \boldsymbol{z}^T \mathrm{diag}(\boldsymbol{\xi})\boldsymbol{z}$ *where* $\boldsymbol{\xi} \quad 0$*;*
- *Uncertain positively weighted sum of convex functions of the form* $(\boldsymbol{\xi}, \boldsymbol{z}) \mapsto \sum_{i \in Q} \xi_i \varphi_i(\boldsymbol{z})$ *with* $\Xi \quad \mathbb{R}^{|U|}_+$*, e.g.,* $(\boldsymbol{\xi}, \boldsymbol{x}, \boldsymbol{y}) \mapsto \sum_{i \in Q} \xi_i x_i^2 / y_i$ *with* $\boldsymbol{y} \quad \boldsymbol{0}$*.*

Example 5 (Violating Assumption L)**.** *We give here some examples of functions which do not satisfy Assumption L.*

- *Non-concave functions of the uncertainty, e.g.,* $(\boldsymbol{\xi}, \boldsymbol{z}) \mapsto \|\boldsymbol{z} \quad \boldsymbol{\xi}\|$ *for any given norm;*
- *General uncertain quadratic form* $(\boldsymbol{\Sigma}, \boldsymbol{z}) \mapsto \boldsymbol{z}^T \boldsymbol{\Sigma} \boldsymbol{z}$ *even with* $\boldsymbol{\Sigma} \quad 0$ *(unless* $\Xi \setminus \mathbb{R}^{|U|} = \emptyset$*)*

In the following lemma, we finally state the class of problems we consider.

Lemma 1. *Under Assumptions* (H)-(L)*, there exists* $[\boldsymbol{l}, \boldsymbol{u}]$ $\mathsf{R}^{jIj+jJj}$ *such that* (3.1) *is equivalent to the following problem:*

$$\inf_{\boldsymbol{x} 2 X \setminus [\boldsymbol{l},\boldsymbol{u}]} \sup_{\boldsymbol{\xi} 2 \Xi} \inf_{(\boldsymbol{t},\boldsymbol{y}) 2 Y^0(\boldsymbol{x})} \sum_{i 2 Q} w_i(\boldsymbol{\xi}) t_i \tag{2SRO-P}$$

with $Y^0(\boldsymbol{x})$ *such that* $Y(\boldsymbol{x}) = \mathrm{proj}_{\boldsymbol{y}}(Y^0(\boldsymbol{x}))$ *and* $\mathrm{cont}\,(Y^0(\boldsymbol{x}))$ *is a convex and nite-dimensional set.*

Proof. The existence of the hyper-rectangle $[\boldsymbol{l}, \boldsymbol{u}]$ is trivial as X is assumed to be bounded (Assumption I.1). Moreover, the following equality holds:

$$\inf_{\boldsymbol{y}} \left\{ \sum_{i 2 Q} w_i(\boldsymbol{\xi}) \rho_i(\boldsymbol{x}, \boldsymbol{y}) : \boldsymbol{y} \; 2 \; Y(\boldsymbol{x}) \right\} = \inf_{\boldsymbol{t},\boldsymbol{y}} \left\{ \sum_{i 2 Q} w_i(\boldsymbol{\xi}) t_i : \boldsymbol{y} \; 2 \; Y(\boldsymbol{x}), t_i = \rho_i(\boldsymbol{x}, \boldsymbol{y}), 8i \; 2 \; Q \right\}$$

However, the optimization problem on the right side of the equality may fail to be convex if there exists $i \; 2 \; Q$ such that ρ_i is not affine. Let Q^A Q be the set of indices for which ρ_i is affine. By Assumption L, for all $i \; 2 \; QnQ^A$, we have $w_i(\,)$ 0 and thus constraint "$t_i = \rho_i(\boldsymbol{x}, \boldsymbol{y})$" may be equivalently replaced by "t_i $\rho_i(\boldsymbol{x}, \boldsymbol{y})$", which is convex. We therefore can choose

$$Y^0(\boldsymbol{x}) = \left\{ (\boldsymbol{t}, \boldsymbol{y}) : \begin{array}{ll} \boldsymbol{y} \; 2 \; Y(\boldsymbol{x}) & \\ t_i = \rho_i(\boldsymbol{x}, \boldsymbol{y}) & 8i \; 2 \; Q^A \\ t_i \quad \rho_i(\boldsymbol{x}, \boldsymbol{y}) & 8i \; 2 \; QnQ^A \end{array} \right\}$$

For every $\boldsymbol{x} \; 2 \; \mathrm{cont}\,(X)$, the continuous relaxation of $Y^0(\boldsymbol{x})$ is convex and non-empty (Assumption J); by construction, it is also finite dimensional. □

In what remains, we will assume to know a hyper-rectangle $[\boldsymbol{l}, \boldsymbol{u}]$ as described in Lemma 1.

3.3 A hull-relaxation-based branch-and-bound algorithm

In this section we present our main contribution and its theoretical foundations for this chapter. We first turn problem (2SRO-P) from a min-max-min problem to a min-max problem in our mixed-integer and convex context. Then, since linear duality does not apply in our setting, we resort to Fenchel duality to obtain a reformulation of the problem. The rest of the development is dedicated to dealing with a non-convex constraint implying the convex hull of a set of points which depend on some decision variables. While in the linear and binary case (see Chapter 2 and Arslan and Detienne [2021]), an exact reformulation could be obtained, we show that one obtains a relaxation by a similar approach in the general case. This relaxation is thus embedded into a branch-and-bound scheme to obtain an optimal solution of (2SRO-P).

3.3.1 Problem reformulation

The following lemma extends to the mixed-integer and convex context the result given in Arslan and Detienne [2021].

Lemma 2 (Single-stage reformulation)**.** *Problem* (2SRO-P) *is equivalent to the following problem:*

$$\inf_{(\boldsymbol{x},\boldsymbol{t},\boldsymbol{y})\in F} \sup_{\boldsymbol{\xi}\in\Xi} \sum_{i\in Q} w_i(\boldsymbol{\xi}) t_i \tag{3.2}$$

with $F = \{(\boldsymbol{x},\boldsymbol{t},\boldsymbol{y}) : \boldsymbol{x} \in X \setminus [\boldsymbol{l},\boldsymbol{u}], (\boldsymbol{t},\boldsymbol{y}) \in \operatorname{conv}(Y^0(\boldsymbol{x}))\}$.

Proof. This lemma relies on the same arguments as those employed in Arslan and Detienne [2021]: first, the feasible space of the inner minimization problem is replaced by its convex hull by linearity of the objective function and convexity of the feasible region. By assumption I.2 and Lemma 1, both Ξ and $\operatorname{conv}(Y^0(\boldsymbol{x}))$ (for all $\boldsymbol{x} \in X$) are convex and finite dimensional. Thus, the result in Perchet and Vigeral [2015] can be used to turn the inner sup inf into an inf sup problem. This achieves the proof. □

The inner maximization problem may be turned into a minimization problem by use of Fenchel duality, as done in Ben-Tal et al. [2014]. In the following proposition, we therefore derive a general reformulation of problem (2SRO-P).

Proposition 1 (Deterministic reformulation)**.** *Problem* (2SRO-P) *is equivalent to the following problem:*

$$\inf_{\boldsymbol{x},\boldsymbol{y},\boldsymbol{t},(\boldsymbol{v}^i)_{i\in Q},\boldsymbol{\xi}} \delta(\boldsymbol{\xi} | \Xi) - \sum_{i\in Q} (t_i w_i)^\top (\boldsymbol{v}^i) \tag{3.3}$$

$$\text{s.t. } \boldsymbol{x} \in X \setminus [\boldsymbol{l},\boldsymbol{u}] \tag{3.4}$$

$$(\boldsymbol{t},\boldsymbol{y}) \in \operatorname{conv}(Y^0(\boldsymbol{x})) \tag{3.5}$$

$$\sum_{i\in Q} \boldsymbol{v}^i = \boldsymbol{\xi} \tag{3.6}$$

$$\boldsymbol{v}^i \in \mathbb{R}^{|U|} \qquad \forall i \in Q \tag{3.7}$$

Proof. By a direct application of Fenchel duality and some conjugate calculus results, the following holds

$$\begin{aligned}
\sup_{\boldsymbol{\xi}\in\Xi} \sum_{i\in Q} t_i w_i(\boldsymbol{\xi}) &= \sup_{\boldsymbol{\xi}\in\mathbb{R}^{|U|}} \left\{ \sum_{i\in Q} t_i w_i(\boldsymbol{\xi}) - \delta(\boldsymbol{\xi} | \Xi) \right\} = \inf_{\boldsymbol{\xi}\in\mathbb{R}^{|U|}} \left\{ \delta(\boldsymbol{\xi} | \Xi) - \left(\sum_{i\in Q} t_i w_i(\boldsymbol{\xi}) \right) \right\} \\
&= \inf_{\boldsymbol{\xi}\in\mathbb{R}^{|U|}} \left\{ \delta(\boldsymbol{\xi} | \Xi) - \sup_{\boldsymbol{v}^i\in\mathbb{R}^{|U|}, i\in Q} \left\{ \sum_{i\in Q} (t_i w_i)^\top (\boldsymbol{v}^i) : \sum_{i\in Q} \boldsymbol{v}^i = \boldsymbol{\xi} \right\} \right\} \\
&= \inf \left\{ \delta(\boldsymbol{\xi} | \Xi) - \sum_{i\in Q} (t_i w_i)^\top (\boldsymbol{v}^i) : \sum_{i\in Q} \boldsymbol{v}^i = \boldsymbol{\xi}, \boldsymbol{v}^i \in \mathbb{R}^{|U|}, i \in Q, \boldsymbol{\xi} \in \mathbb{R}^{|U|} \right\}
\end{aligned}$$

See also appendix A for more details on conjugate calculus. □

Remark 9. *Assume wlog that* $|Q| = |U|$*. If, for all* $i \in Q$*,* $w_i(\boldsymbol{\xi}) = w_i(\xi_i)$*, then problem* (2SRO-P) *is equivalent to*

$$\inf_{(\boldsymbol{x},\boldsymbol{t},\boldsymbol{y})\in F} \left\{ \delta(\boldsymbol{\xi} | \Xi) - \sum_{i\in Q} (t_i w_i)^\top (\boldsymbol{\xi}) \right\} \tag{3.8}$$

Remark 10. *Let* $i \in Q$ *such that* $w_i(\,)$ *is a ne, i.e.,* $w_i(\boldsymbol{\xi}) = (\boldsymbol{r}^i)^T \boldsymbol{\xi} + r_{i0}$*. Problem* (2SRO-P) *is*

equivalent to

$$\inf_{(\boldsymbol{x},\boldsymbol{t},\boldsymbol{y})\in F} \left\{ \delta\ (\boldsymbol{R}\boldsymbol{t}j\Xi) + \boldsymbol{r}_0^T \boldsymbol{t} \right\} \tag{3.9}$$

Proof. Indeed, we have

$$(t_i w_i)\ (\boldsymbol{v}) = \inf_{\boldsymbol{\xi}\in \mathsf{R}^{jUj}} f\boldsymbol{v}^T\boldsymbol{\xi} \quad t_i((\boldsymbol{r}^i)^T\boldsymbol{\xi} + r_{i0})g = \begin{cases} t_i r_{i0} & \text{if } \boldsymbol{v} = t_i \boldsymbol{r}^i \\ 1 & \text{otherwise.} \end{cases}$$

□

These results show that although the reformulation for the general case adds *jQj jUj* continuous variables, for some relevant cases these additional variables can be omitted. In particular this is true in case all the $w_i(\ )$ functions are either separable or affine.

3.3.2 Relaxation

Note that the deterministic reformulation presented above still is not, in general, a convex MINLP and that no tractable, compact form is known in the general case. To overcome this drawback, we replace constraint $(\boldsymbol{t},\boldsymbol{y})\ 2\ \mathrm{conv}\,(Y^0(\boldsymbol{x}))$ (3.5) by the following relaxed requirement:

$$(\boldsymbol{x},\boldsymbol{t},\boldsymbol{y})\ 2\ \mathrm{conv}\,(S)\ \text{ with } S = \left\{ (\boldsymbol{x},\boldsymbol{t},\boldsymbol{y}) : \begin{array}{llll} l_j & x_j & u_j & 8j\ 2\ I \\ x_j\ 2\ \mathsf{Z} & & & 8j\ 2\ I_I \\ (\boldsymbol{t},\boldsymbol{y})\ 2\ Y^0(\boldsymbol{x}) & & & \end{array} \right\}. \tag{3.10}$$

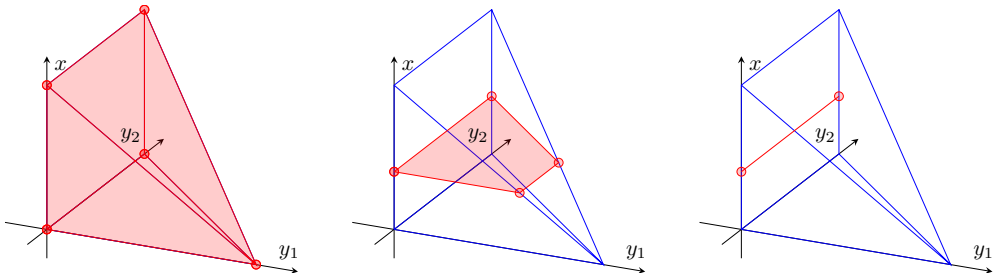
The problem obtained from this substitution is thus

$$\begin{array}{rl} \min\limits_{\boldsymbol{x},\boldsymbol{y},(\boldsymbol{v}^i)_{i2Q},\boldsymbol{\xi}} & \delta\ (\boldsymbol{\xi}j\Xi) \quad \sum\limits_{i2Q}(t_i w_i)\ \left(\boldsymbol{v}^i\right) \\ \text{s.t.} & \boldsymbol{x}\ 2\ X \setminus [\boldsymbol{l},\boldsymbol{u}] \\ & (\boldsymbol{x},\boldsymbol{t},\boldsymbol{y})\ 2\ \mathrm{conv}\,(S) \\ & \sum\limits_{i2Q} \boldsymbol{v}^i = \boldsymbol{\xi} \\ & \boldsymbol{v}^i\ 2\ \mathsf{R}^{jUj} \qquad 8i\ 2\ Q \\ & \boldsymbol{\xi}\ 2\ \mathsf{R}^{jUj} \end{array} \tag{P}$$

It is clear that, for any fixed $\bar{\boldsymbol{x}}\ 2\ X$, we have $f\bar{\boldsymbol{x}}g \quad Y^0(\bar{\boldsymbol{x}}) = S \setminus f(\boldsymbol{x},\boldsymbol{t},\boldsymbol{y}) : \boldsymbol{x} = \bar{\boldsymbol{x}}g$, and that the same holds even for $\bar{\boldsymbol{x}}\ 2\ \mathrm{cont}\,(X)$. However, as shown, e.g., in Sherali and Zhu [2006], the convexified counterpart does not hold, in the sense that the inclusion "$f\bar{\boldsymbol{x}}g \quad \mathrm{conv}\,(Y(\bar{\boldsymbol{x}}))$ $\mathrm{conv}\,(S) \setminus f(\boldsymbol{x},\boldsymbol{t},\boldsymbol{y}) : \boldsymbol{x} = \bar{\boldsymbol{x}}g$" may be strict. Example 6 below illustrates this case.

Example 6 (Hull relaxation)**.** *We consider the rst- and second-stage feasible sets introduced in Example 3. In Figure* (3.1a)*, we represent the convex hull of* S*. For a xed rst-stage decision* $\bar{x}$ *(here,* $\bar{x} = 0.4$*), Figure* (3.1b) *reports the feasible points for constraint* (3.10)*, whereas Figure* (3.1c) *describes the exact shape of* $\mathrm{conv}\,(Y(\bar{x}))$*. The gure shows an example in which inclusion is strict. In addition, note that, whenever* $\bar{x}$ *attains its bounds (i.e.,* $\bar{x}\ 2\ f0,1g$*),* $f\bar{x}g \quad \mathrm{conv}\,(Y(\bar{x})) = \mathrm{conv}\,(S) \setminus f(x,\boldsymbol{y}) : x = \bar{x}g$ *holds.*

The following lemma follows from the considerations above.



(a) conv (S) (b) conv (S) \ $fx = xg$ (c) fxg conv $(Y(x))$

Figure 3.1: Graphical representation of different sets from example 3

Lemma 3 (Lower-bounding property)**.** *Denoting by* $v(\ )$ *the optimal objective value of problem , we have:*

$$v(\mathrm{P}) \quad v(\text{2SRO-P})$$

In other words, (P) is a relaxation of (2SRO-P). In the next proposition, we introduce a condition under which a feasible solution for problems (P) is feasible for problem (2SRO-P) as well.

Proposition 2. *If* $\bar{\boldsymbol{x}}$ *2* vert $([\boldsymbol{l}, \boldsymbol{u}])$*, then*

$$f\bar{\boldsymbol{x}}g \quad \text{conv}\,(Y^0(\bar{\boldsymbol{x}})) = \text{conv}\,(S) \setminus f(\boldsymbol{x}, \boldsymbol{t}, \boldsymbol{y}) : \boldsymbol{x} = \bar{\boldsymbol{x}}g$$

Proof. Let $\bar{\boldsymbol{x}}$ *2* vert $([\boldsymbol{l}, \boldsymbol{u}])$ and let $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{t}}, \hat{\boldsymbol{y}})$ *2* conv (S) \ $f(\boldsymbol{x}, \boldsymbol{t}, \boldsymbol{y}) : \boldsymbol{x} = \bar{\boldsymbol{x}}g$. Then, $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{t}}, \hat{\boldsymbol{y}})$ can be expressed as a (finite) convex combination of points of conv (S) (Carathéodory's theorem), i.e.,

$$(\hat{\boldsymbol{x}}, \hat{\boldsymbol{t}}, \hat{\boldsymbol{y}}) = \sum_{e2E} (\bar{\mathbf{x}}^e, \bar{\mathbf{t}}^e, \bar{\mathbf{y}}^e)\alpha_e$$

where E is a given index list of such elements of conv (S). Assume that there exists j *2* I and i *2* E such that $\bar{\mathrm{x}}^i_j$ *6=* $\bar{x}_j$. If $\bar{\mathrm{x}}^i_j > \bar{x}_j$, condition $\bar{\mathbf{x}}^i$ *2* conv (S) implies that $\bar{x}_j = l_j$. Hence, $\alpha_i = 0$ since $\bar{\mathrm{x}}^k_j \quad l_j$ *8*k *2* E. The same argument shows that $\bar{\mathrm{x}}^i_j < \bar{x}_j$ implies $\alpha_i = 0$. Thus, for each e *2* E such that $\alpha_e > 0$, we must have $\bar{\mathbf{x}}^e = \bar{\boldsymbol{x}}$. This implies that $(\bar{\mathbf{t}}^e, \bar{\mathbf{y}}^e)$ *2* $Y^0(\bar{\boldsymbol{x}})$ and thus $\sum_{e2E}(\bar{\mathbf{t}}^e, \bar{\mathbf{y}}^e)\alpha_e$ *2* conv $(Y^0(\bar{\boldsymbol{x}}))$. □

Corrolary 1 (Tightness condition)**.** *Let* X *be the set of optimal rst-stage decisions of problem* (P)*. Then:*

$$X \setminus \text{vert}\,([\boldsymbol{l}, \boldsymbol{u}]) \text{ 6= } ; \ ) \ v(\mathrm{P}) = v(\text{2SRO-P})$$

Proof. Let $(\boldsymbol{x}^*, \boldsymbol{t}^*, \boldsymbol{y}^*)$ be an optimal solution of (P) with $\boldsymbol{x}^*$ *2* vert $([\boldsymbol{l}, \boldsymbol{u}])$. From Proposition 2, it is also feasible for problem (2SRO-P). Thus, Lemma 3 implies optimality for problem (2SRO-P). □

This result directly implies Corollary 2 which states that, in the special case where the first-stage variables are all binary, problem (P) is always an exact reformulation of (2SRO-P).

Corrolary 2 (Tightness condition/binary case)**.** *If the first-stage decisions are all binary, i.e.,* $I_C = \emptyset$ *and* $[\boldsymbol{l}, \boldsymbol{u}] = [\mathbf{0}, \mathbf{1}]$*, then*

$$v(\mathrm{P}) = v(\text{2SRO-P})$$

Proof. Any optimal first-stage solution $\boldsymbol{x}$ satisfies $\boldsymbol{x} \in \{0,1\}^{|I_I|} = \mathrm{vert}([\boldsymbol{l}, \boldsymbol{u}])$ which, by Corollary 1, proves the result. □

3.3.3 Enumerative algorithm

We now present an exact method for solving problem (2SRO-P). Motivated by Corollary 1, the main idea of the algorithm is to determine an optimal value of the first-stage variables, and then derive the corresponding optimal values for the second-stage variables. To this aim, we developed a branch-and-bound algorithm in which we relax both the integrality of the $\boldsymbol{x}$ and requirement (3.5). To ensure feasibility, we perform a spatial branching on the $\boldsymbol{x}$ variables, until each of its components attains either its lower or upper bound. The algorithm stores the best feasible solution found (the *incumbent* solution) which is returned when the algorithm stops.

Node solution

Let p denote a generic node of the branch-and-bound tree, associated with bounds $\boldsymbol{l}^p$ and $\boldsymbol{u}^p$ on first-stage variables.

A lower bound on the optimal solution value of node p can be computed solving the following problem:

$$\begin{aligned} \min_{\boldsymbol{x},\boldsymbol{t},\boldsymbol{y},(\boldsymbol{v}^i)_{i\in Q},\boldsymbol{\xi}} \quad & \delta(\boldsymbol{\xi}|\Xi) \sum_{i\in Q}(t_i w_i) \quad (\boldsymbol{v}^i) \\ \text{s.t.} \quad & \boldsymbol{x} \in \mathrm{cont}(X) \setminus [\boldsymbol{l}^p, \boldsymbol{u}^p] \\ & (\boldsymbol{x}, \boldsymbol{t}, \boldsymbol{y}) \in \mathrm{conv}(S^p) \\ & \sum_{i\in Q} \boldsymbol{v}^i = \boldsymbol{\xi} \\ & \boldsymbol{v}^i \in \mathbb{R}^{|U|} \qquad \forall i \in Q \\ & \boldsymbol{\xi} \in \mathbb{R}^{|U|} \end{aligned} \tag{LB^p}$$

where $S^p = \{(\boldsymbol{x}, \boldsymbol{t}, \boldsymbol{y}) : \boldsymbol{l}^p \le \boldsymbol{x} \le \boldsymbol{u}^p,\ x_j \in \mathbb{Z}, \forall j \in I_I,\ (\boldsymbol{t}, \boldsymbol{y}) \in Y^0(\boldsymbol{x})\}$. This problem is exactly the continuous relaxation of problem (P) where the bounds $\boldsymbol{l}$ and $\boldsymbol{u}$ have been replaced by $\boldsymbol{l}^p$ and $\boldsymbol{u}^p$. Note that at the root node we have $\boldsymbol{l}^0 = \boldsymbol{l}$ and $\boldsymbol{u}^0 = \boldsymbol{u}$.

Let $(\boldsymbol{x}^p, \boldsymbol{t}^p, \boldsymbol{y}^p, (\boldsymbol{v}^{ip})_{i\in Q}, \boldsymbol{\xi}^p)$ be an optimal solution of problem LB^p. If $v(\mathrm{LB}^p)$ is greater than or equal to the cost of the incumbent, the node is fathomed by bounding. Otherwise, we distinguish three cases:

- if $\boldsymbol{x}^p \in \mathrm{vert}([\boldsymbol{l}^p, \boldsymbol{u}^p])$, by Proposition 2, this solution is optimal for the current node. Hence, the node is fathomed by optimality and the incumbent is updated;
- if $\boldsymbol{x}^p \in X \cap \mathrm{vert}([\boldsymbol{l}^p, \boldsymbol{u}^p])$, we compute a feasible solution for (2SRO-P) by solving the following model in which the first-stage variables are fixed to $\boldsymbol{x}^p$:

$$
\begin{aligned}
\min_{\boldsymbol{t},\boldsymbol{y},(\boldsymbol{v}^i)_{i\,2\,Q},\boldsymbol{\xi}} \quad & \delta\ (\boldsymbol{\xi} j \Xi) \quad \sum_{i\,2\,Q}(t_i w_i)\ \ (\boldsymbol{v}^i) \\
\text{s.t.} \quad & (\boldsymbol{t},\boldsymbol{y})\ 2\ \text{conv}\,(Y^0(\boldsymbol{x}^p\ )) \\
& \sum_{i\,2\,Q} \boldsymbol{v}^i = \boldsymbol{\xi} \qquad\qquad (\text{UB}^p) \\
& \boldsymbol{v}^i\ 2\ \mathsf{R}^{jUj} \qquad 8i\ 2\ Q \\
& \boldsymbol{\xi}\ 2\ \mathsf{R}^{jUj}
\end{aligned}
$$

Note that, in this case, $\boldsymbol{x}^p$ corresponds to a feasible first-stage solution; hence, by Assumption J, problem UB^p is always feasible, and possibly the incumbent is updated. If $v(LB^p) = v(UB^p)$ then node p is solved; otherwise, we perform a branching;

– if $\boldsymbol{x}^p\ \ 2\ \text{cont}\,(X)\ n\ X$, we branch.

In the last case, before branching, one can try to round $\boldsymbol{x}^p$; if the resulting point is in X, a feasible solution for (2SRO-P) can be computed. In our experiments, every fractional value for x_j^p with $j\ 2\ I_I$ was rounded to the closest integer while variables x_j^p with $j\ 2\ I_C$ were not rounded.

Branching

We now describe how to select the branching variable at node p. For each first-stage variable, say with index $j\ 2\ I$, we compute the minimum distance of x_j^p from one of its bounds at the node, i.e., we evaluate:

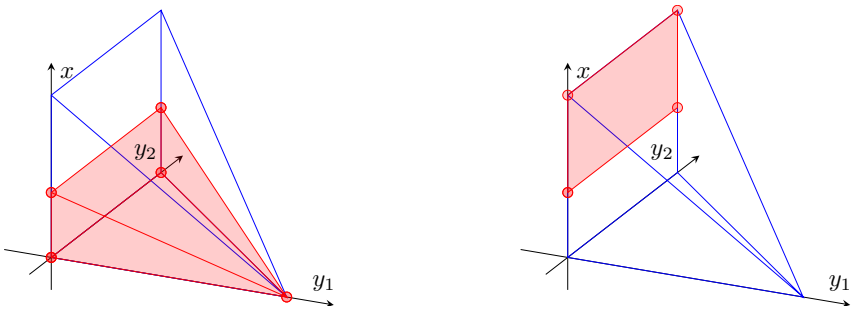
$$\theta_j^p = \min f x_j^p \qquad l_j^p; u_j^p \qquad x_j^p\ g.$$

For branching, we give priority to integer variables that do not attain their bound. Otherwise, we resort to spatial branching on continuous variables. In both cases, we select the variable with maximum θ_j^p value, i.e., we select variable $x_{\bar{j}}$ such that,

$$\bar{j}\ 2\ \begin{cases} \text{argmax} f \theta_j^p : j\ 2\ I_I g & \text{if } 9j\ 2\ I_I, \theta_j > 0 \\ \text{argmax} f \theta_j^p : j\ 2\ I_C g & \text{otherwise.} \end{cases}$$

If $\bar{j}\ 2\ I_I$, then a standard integer branching is executed. Otherwise, we resort to spatial branching, and generate two descendant nodes by imposing $x_{\bar{j}} \quad x_{\bar{j}}^p$ for the left node and $x_{\bar{j}} \quad x_{\bar{j}}^p$ for the right one. We associate to each node the lower bound value of the current node $v(\text{LB}^p)$ and insert them in a list of open nodes. At each iteration, we extract from the list one node with minimum lower bound value, halting the algorithm when the list is empty.

Example 7. *Figure 3.2 illustrates the left and right child obtained by spatial branching on* $x \quad \beta$ *and* $x \quad \beta$ *from example 3 (here,* $\beta = 0.4$*). Clearly, the right child allows the same recourse decisions as in* $Y(x)$ *for all* $x \quad \beta$*. The left child, however, still allows second-stage decisions that could end up being infeasible in the original problem. In particular,* $(\boldsymbol{x},\boldsymbol{y}) = (\varepsilon, 1 \quad \varepsilon, 0)$ *with* $\varepsilon\ 2\ (0,\beta]$ *is feasible for* (LB^p) *but not for* (2SRO-P).



(a) Left child (x β) (b) Right child (x β)

Figure 3.2: Branching on continuous variable x from example 3

Convergence

In the following, we analyze the convergence of the branch-and-bound algorithm. While finite convergence is ensured if all first-stage variables are integer, this may not be the case when the first-stage includes continuous variables. We now consider the case where our branch-and-bound algorithm has an infinite number of nodes. Note that, in this case, there exists at least one infinite branch to the branch-and-bound tree since the number of variables which can be selected for branching is finite. We consider one such branch and denote it by P. For each node p 2 P, we denote by $(\boldsymbol{l}^p, \boldsymbol{u}^p)$ the associated bounds for the $\boldsymbol{x}$ variables and by $(\boldsymbol{x}^p, \boldsymbol{t}^p, \boldsymbol{y}^p, \boldsymbol{V}^p, \boldsymbol{\xi}^p)$ an optimal solution to the corresponding lower-bounding problem. Additionally, we introduce functions f_{LB} and f_{ST} which correspond to the objective function of the lower bounding problem and of the single stage reformulation from Lemma 2, respectively. For the reader's convenience, we recall them here.

$$f_{LB}(\boldsymbol{t}, \boldsymbol{V}, \boldsymbol{\xi}) := \delta\ (\boldsymbol{\xi} j \Xi) \quad \sum_{i 2 Q} (t_i w_i)\ \ (\boldsymbol{v}^i) \tag{3.11}$$

$$f_{ST}(\boldsymbol{t}) := \sup_{\boldsymbol{\xi} 2 \Xi} \sum_{i 2 Q} w_i(\boldsymbol{\xi}) t_i \tag{3.12}$$

Remark 11. *For each node p 2 P it holds $f_{LB}(\boldsymbol{t}^p, \boldsymbol{V}^p, \boldsymbol{\xi}^p) = f_{ST}(\boldsymbol{t}^p)$.*

Proof. This directly follows from the definition of $(\boldsymbol{x}^p, \boldsymbol{t}^p, \boldsymbol{y}^p, \boldsymbol{V}^p, \boldsymbol{\xi}^p)$ and Proposition (1). □

Lemma 4. *Let P be a sequence of nodes of any in nite branch of the branch-and-bound tree. Then,*

(i) The sequence $f(\boldsymbol{l}^p, \boldsymbol{u}^p)g_{p 2 P}$ has a unique accumulation point, which we denote by $(\boldsymbol{l}, \boldsymbol{u})$;

(ii) The sequence $f(\boldsymbol{x}^p, \boldsymbol{t}^p, \boldsymbol{y}^p)g_{p 2 P}$ has at least one accumulation point;

(iii) Let $\boldsymbol{x}$ be any accumulation point of $f\boldsymbol{x}^p\ g_{p 2 P}$, then, for each j 2 I_C which is in nitely selected for branching, there exists a sub-sequence P^j P such that either $f u_j^p g_{p 2 P^j}$! x_j or $f l_j^p g_{p 2 P^j}$! x_j;

(iv) Every accumulation point $\boldsymbol{x}$ of $f\boldsymbol{x}^p\ g_{p 2 P}$ satis es $\boldsymbol{x}$ 2 vert $([\boldsymbol{l}, \boldsymbol{u}])$.

Proof.

(i) This follows from the fact that $\boldsymbol{l}^p$ (resp. $\boldsymbol{u}^p$) is a bounded, non-decreasing (resp. non-increasing) sequence.

(ii) This follows from the Bolzano-Weierstrass theorem since the sequence $\{\boldsymbol{x}^p\}_{p \in P}$ is generically bounded by $[\boldsymbol{l}, \boldsymbol{u}]$, $\overline{X}$ is compact and $\mathrm{conv}(S)$ is closed and bounded, thus compact (indeed, for all $i \in Q$, t_i^p is trivially bounded by $\sup\{\varphi_i(\boldsymbol{x}, \boldsymbol{y}) : \boldsymbol{x} \in \overline{X}, \boldsymbol{y} \in \overline{Y}(\boldsymbol{x})\}$ which is finite by Assumption K.1).

(iii) Consider any accumulation point $\boldsymbol{x}$ of $\{\boldsymbol{x}^p\}_{p \in P}$ with its associated convergent sub-sequence $P' \subseteq P$, i.e., $\{\boldsymbol{x}^p\}_{p \in P'} \to \boldsymbol{x}$. Let $j \in I_C$ be as described in the lemma, and consider the sub-sequence $P^{u_j} \subseteq P'$ such that, for all $p \in P^{u_j}$, $u_j^{p+1} = x_j^p$. Assume P^{u_j} is not finite. Then, we have that $\{x_j^p\}_{p \in P^{u_j}} \to x_j$ since $P^{u_j} \subseteq P'$. And thus, by definition of P^{u_j}, we have that $\{u_j^{p+1}\}_{p \in P^{u_j}} \to x_j$. We therefore chose $P^j = \{p+1 : p \in P^{u_j}\}$ and have $\{u_j^p\}_{p \in P^j} \to x_j$. If instead P^{u_j} is finite, the sub-sequence $P^{l_j} \subseteq P'$ defined by nodes p for which $l_j^{p+1} = x_j^p$ is infinite; therefore, the similar argument can be applied.

(iv) We have just shown that, for any accumulation point $\boldsymbol{x}$ of $\{\boldsymbol{x}^p\}_{p \in P}$, with its associated convergent sub-sequence $P' \subseteq P$, and any infinitely branched index $j \in I_C$, there exists $P^j \subseteq P'$ such that either $\{u_j^p\}_{p \in P^j} \to x_j$ or $\{l_j^p\}_{p \in P^j} \to x_j$. Assume $\{l_j^p\}_{p \in P^j} \to x_j$ holds. Then, we have that $P^j \subseteq P'$ and $\{l_j^p\}_{p \in P^j} \to l_j$. Thus, $x_j = l_j$ holds, since any sub-sequence of a converging sequence converges to the same point. The same argument can be applied when $\{u_j^p\}_{p \in P^j} \to x_j$.

□

Theorem 2. *Let P be a sequence of nodes of any infinite branch of the branch-and-bound tree. Then, every accumulation point of $\{(\boldsymbol{x}^p, \boldsymbol{t}^p, \boldsymbol{y}^p)\}_{p \in P}$, say $(\boldsymbol{x}, \boldsymbol{t}, \boldsymbol{y})$, is an optimal solution of problem* (3.2)*, and, thus, $\boldsymbol{x}$ is an optimal solution of* (2SRO-P).

Proof. By Lemma 4 (ii), there exists a sub-sequence $P' \subseteq P$ such that $\{(\boldsymbol{x}^p, \boldsymbol{t}^p, \boldsymbol{y}^p)\}_{p \in P'} \to (\boldsymbol{x}, \boldsymbol{t}, \boldsymbol{y})$. Note that $\overline{X}$ and $\mathrm{conv}\,(S)$ are compact sets, hence we have that $(\boldsymbol{x}, \boldsymbol{t}, \boldsymbol{y}) \in \overline{X} \times \mathrm{conv}\,(S)$. Moreover, by Lemma 4 (iv), we know that $\boldsymbol{x} \in \mathrm{vert}\,([\boldsymbol{l}, \boldsymbol{u}])$ which, by Proposition 2, ensures that $(\boldsymbol{t}, \boldsymbol{y}) \in \mathrm{conv}\,(Y^0(\boldsymbol{x}))$. Hence, $(\boldsymbol{x}, \boldsymbol{t}, \boldsymbol{y})$ is feasible for (3.2). Note that f_{ST} is a continuous function since it is the point-wise supremum of continuous (affine) functions. Thus, by Remark 11, we have $\{f_{LB}(\boldsymbol{t}^p, \boldsymbol{V}^p, \boldsymbol{\xi}^p)\}_{p \in P'} \to f_{ST}(\boldsymbol{t})$. In other words, the objective value of the feasible solution $(\boldsymbol{x}, \boldsymbol{t}, \boldsymbol{y})$ to (3.2) is $f_{ST}(\boldsymbol{t})$. Yet, by Lemma 2, we know that (3.2) and (2SRO-P) have the same objective value. This makes $\boldsymbol{x}$ a feasible solution to (2SRO-P) of value $f_{ST}(\boldsymbol{t})$. Since our node selection strategy always picks a node with minimum lower bound, for each node $p \in P$, we have $f_{LB}(\boldsymbol{t}^p, \boldsymbol{V}^p, \boldsymbol{\xi}^p) = v(LB^p) \leq v(\text{2SRO-P}) \leq f_{ST}(\boldsymbol{t})$. As $v(LB^p)$ converges to $f_{ST}(\boldsymbol{t})$, we also have $f_{LB}(\boldsymbol{t}^p, \boldsymbol{V}^p, \boldsymbol{\xi}^p) = f_{ST}(\boldsymbol{t}) = v(\text{2SRO-P})$. □

Thus, asymptotic convergence of our branch-and-bound algorithm is proved. We further give a sufficient condition for finite ε-convergence even when $I_C \neq \emptyset$. More specifically, given an optimal solution $\boldsymbol{x}^p$ of the lower bounding problem at a node p, we introduce the following function

$$G(\boldsymbol{x}^p) := v(\mathrm{UB}^p),$$

that returns the solution value in the upper bounding problem. We show that, if G is a continuous function, then our branch-and-bound algorithm enjoys finite ε-convergence.

Proposition 3 (Sufficient condition for finite ε-convergence)**.** *Let $\varepsilon > 0$ be a given precision and assume that G is continuous. Then our branch-and-bound algorithm converges to an ε-optimal solution of* (2SRO-P) *in a finite number of iterations.*

Proof. By continuity of G, we have $\{G(\boldsymbol{x}^p)\}_{p \in P^0} \to G(\boldsymbol{x}) = v(\text{LB})$, which allows us to fathom the node by optimality after a finite number of nodes for any positive tolerance. □

We conclude this section by observing that, at each node of the branch-and-bound algorithm, the lower bounding problem can be solved with ε-tolerance in a finite number of operations. Indeed, as shown in Ceria and Soares [1999] and Grossmann and Ruiz [2011], one can reformulate a convex disjunctive program as a compact convex MINLP by introducing an exponential number of auxiliary variables that model the disjunctions. The resulting model can thus be solved in finite number of states by using any algorithm designed for convex optimization.

Example 8 (Asymptotic convergence)**.** *We further elaborate on Example 3 by exhibiting a case in which our algorithm may not finitely converge but, instead, has asymptotic convergence in the first-stage solution values. Remember that, in this case, Problem* (2SRO-P) *is equivalent to solving "$\min_{x \in [0,1]} G(x)$ with $G(x) = \mathbb{1}(x > 0)$". Applying the reformulation from Lemma 2, one obtains the following reformulation of G.*

$$\begin{aligned} G(x) = \min \ & \max\{0, \quad 2y_1 + y_2 + 1\} \\ & \text{s.t. } (y_1, y_2) \in \text{conv}\,(Y(x)) \end{aligned}$$

Applying a hull-relaxation, one obtains that "$\min_{x \in [0,1]} f_{LB}(x)$" with f_{LB} defined as follows, is a lower-bounding problem for (2SRO-P)*.*

$$f_{LB}(x) = \min \ \max\{0, \quad 2y_1 + y_2 + 1\} \tag{3.13}$$

$$\text{s.t. } (x, y_1, y_2) \in \text{conv}\left(\left\{(x^0, y_1^0, y_2^0) : \begin{matrix} l & x^0 & u \\ (y_1^0, y_2^0) & \in & Y(x) \end{matrix}\right\}\right) \tag{3.14}$$

In Figure 3.3, we have depicted functions G (in green) and f_{LB} (in black). Now, observe that, for any $u > 0$ and $l = 0$, $x = u/2$ is a minimizer of f_{LB}. Indeed, we always have the following:

$$\begin{pmatrix} u/2 \\ 1/2 \\ 0 \end{pmatrix} = (1/2) \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} + (1/2) \begin{pmatrix} u \\ 0 \\ 0 \end{pmatrix} \text{ and } 1/2 + 1/2 = 1$$

and $f_{LB}(u/2) = \max\{0, \quad 2(1/2) + 0 + 1\} = 0$. However, we have $G(u/2) = 1$, since $u > 0$. Thus, the lower-bounding problem and the upper-bounding problem are always distanced by 1 if $x = u/2$ is returned when minimizing f_{LB} (note that 0 is also a minimize of f_{LB}, if this minimizer is returned, instead, the algorithm finitely terminates).

Now, since $l < u$, our algorithm will branch on x in order to reduce the gap between l and u. Since $u/2$ is always a minimizer of the lower-bounding problem when $u > 0$, the same situation could repeat in nitely while u decreases. Yet, observe that the following holds.

$$\begin{pmatrix} u/2^p \\ 1/2 \\ 0 \end{pmatrix} \; ! \; \begin{pmatrix} 0 \\ 1/2 \\ 0 \end{pmatrix} \text{ as } p \; ! \; +1$$

In other words, the optimal solutions $(x^p\ , y_1^p\ , y_2^p\ )$ of the lower-bounding problem will converge to $(x\ , y_1, y_2) = (0, \frac{1}{2}, 0)$. Clearly, $(0,0)$ 2 $Y(0)$ and $(1,0)$ 2 $Y(0)$, thus $(1/2, 0) = (1/2)(0,0) + (1/2)(1,0)$ 2 $\text{conv}\,(Y(0))$. Thus $(x\ , y_1, y_2)$ is feasible for the reformulation in Lemma (2) and the corresponding objective value is zero. We well have $f_{LB}(x\ ) = G(x\ ) = 0$.

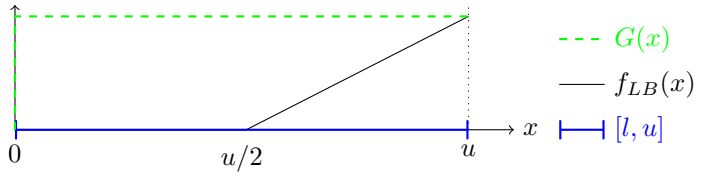


Figure 3.3: Graphical representation of f_{LB} and G

3.3.4 A convexi cation scheme based on column-generation

In this section, we propose a nonlinear column-generation algorithm to be used, at each node p, to solve problem (LB^p) to ε-optimality in a finite number of iterations. According to this scheme, we approximate $\text{conv}\,(S^p)$ by the convex hull of a finite set of points belonging to S^p.

Restricted Master Problem: To determine this set, we use an iterative approach. At each iteration k, let $K = f1, \ldots, kg$ and denote by $H^{pk} = f(\bar{\mathbf{x}}^{pj}, \bar{\mathbf{t}}^{pj}, \bar{\mathbf{y}}^{pj}) : j\ 2\ Kg$ the associated set of points. We clearly have $\text{conv}\left(H^{pk}\right)$ $\text{conv}\,(S^p)$, thus the optimal solution of the problem obtained by substituting $\text{conv}\,(S^p)$ with $\text{conv}\left(H^{pk}\right)$ in (LB^p) gives an upper bound of (LB^p). The

resulting problem, denoted as ($\widehat{\text{LB}}^{pk}$), is called the *Restricted Master*, and is formulated as follows:

$$\min_{\boldsymbol{x},\boldsymbol{t},\boldsymbol{y},\boldsymbol{V},\boldsymbol{\xi},\boldsymbol{\alpha}} \delta\ (\boldsymbol{\xi} j \Xi) \quad \sum_{i 2 Q}(t_i w_i)\ \ (\boldsymbol{v}^i) \tag{3.16}$$

$$\text{s.t. } \boldsymbol{x}\ 2\ \text{cont}\,(X)\ \backslash\ [\boldsymbol{l}^p, \boldsymbol{u}^p] \tag{3.17}$$

$$\boldsymbol{x} = \sum_{j 2 K} \alpha_j \bar{\mathbf{x}}^{pj} \tag{3.18}$$

$$\boldsymbol{t} = \sum_{j 2 K} \alpha_j \bar{\mathbf{t}}^{pj} \tag{3.19}$$

$$\boldsymbol{y} = \sum_{j 2 K} \alpha_j \bar{\mathbf{y}}^{pj} \tag{3.20}\ (\widehat{\text{LB}}^{pk})$$

$$\sum_{j 2 K} \alpha_j = 1 \tag{3.21}$$

$$\sum_{i 2 Q} \boldsymbol{v}^i = \boldsymbol{\xi} \tag{3.22}$$

$$\boldsymbol{v}^i\ 2\ \mathsf{R}^{jUj} \qquad 8i\ 2\ Q \tag{3.23}$$

$$\boldsymbol{\xi}\ 2\ \mathsf{R}^{jUj} \tag{3.24}$$

$$\alpha_j \quad 0 \qquad 8j\ 2\ K \tag{3.25}$$

Following the classical column-generation framework, the current approximation can be improved by means of a so-called *Pricing Problem*, defined as follows.

Pricing Problem: Let $\boldsymbol{\lambda}^{pk}$, $\boldsymbol{\mu}^{pk}$, $\boldsymbol{\pi}^{pk}$ and η^{pk} be the values of the dual variables associated with constraints (3.18), (3.19), (3.20), and (3.21) in an optimal solution of problem ($\widehat{\text{LB}}^{pk}$). Pricing asks to solve the following problem

$$(\bar{\mathbf{x}}^{p,k+1}, \bar{\mathbf{t}}^{p,k+1}, \bar{\mathbf{y}}^{p,k+1})\ 2\ \underset{(\boldsymbol{x},\boldsymbol{t},\boldsymbol{y})\,2\,S^p}{\text{argmin}} \quad \boldsymbol{\lambda}^{pk\ T}\boldsymbol{x} \quad \boldsymbol{\mu}^{pk^T}\boldsymbol{t} \quad \boldsymbol{\pi}^{pk\ T}\boldsymbol{y} \quad \eta^{pk\ T} \tag{PP^{pk}}$$

and generates a new point $(\bar{\mathbf{x}}^{p,k+1}, \bar{\mathbf{t}}^{p,k+1}, \bar{\mathbf{y}}^{p,k+1})$ belonging to S^p. If $v(\text{PP}^{pk})$ ε, we have an ε-optimal solution to (LB^p), and hence the algorithm terminates. Otherwise, we set $H^{k+1} = H^k\ [\ f(\bar{\mathbf{x}}^{p,k+1}, \bar{\mathbf{t}}^{p,k+1}, \bar{\mathbf{y}}^{p,k+1})g$, $k = k+1$ and iterate. Note that, at each iteration k, a lower bound on the optimal solution value of (LB^p) is given by $v(\widehat{\text{LB}}^{pk})$ $v(\text{PP}^{pk})$. This lower bound, combined with an upper bound, can allow us to early terminate the solution of problem (LB^p).

The convergence of nonlinear column generation has been established in García et al. [2003].

3.4 Computational experiments

In this section, we report computational results of our solution algorithm when applied to an uncertain counterpart of a Capacitated Facility Location Problem with congestion.

3.4.1 Problem de nition

We consider a variant of the Facility Location Problem, in which we are given a set V_1 of candidate sites for opening facilities, as well as a set V_2 of clients to be served with some product. Each client $j \in V_2$ has a demand d_j representing the quantity of product that she/he wants to receive. Each site $i \in V_1$ can be activated at a given fixed cost $f_i > 0$. In this case, one has to decide the capacity to be installed, at cost u_i per unit of capacity. Each site i has an upper bound $\bar{q}_i$ on the maximum capacity that can be installed. Each connection $(i, j) \in V_1 \quad V_2$ is associated with a fixed cost c_{ij}, and a variable cost t_{ij} per unit of product which is transported. In our setting, we explicitly model congestion at each site i by means of an additional cost which depends on the total amount of product, say v_i, leaving the facility. As in the congested Facility Location Problem considered in Desrochers et al. [1995] and in Fischetti et al. [2016], the congestion cost for site i is given by

$$F_i(v_i) = (\alpha_i + \beta_i v_i^{\gamma_i})v_i \tag{3.25}$$

where $\alpha_i \quad 0$, $\beta_i > 0$ and $\gamma_i \quad 1$ are input parameters. Note that each function F_i is convex for non-negative arguments v_i. The problem asks to determine the capacity to be installed at each opened facility and the flow of product from facilities to clients, so as to serve all clients at minimum cost. This problem can be reduced to the one addressed in Desrochers et al. [1995] in case there are no capacity constraints at the sites (i.e., $\bar{q}_i = \mathbf{1}$ and $u_i = 0$ for each $i \in V_1$) and transportation costs only include a variable component (i.e., $c_{ij} = 0$ for each $(i, j) \in V_1 \quad V_2$).

In our context, connection costs are not known when deciding the capacities to be installed. Formally, for each connection $(i, j) \in V_1 \quad V_2$, we denote by $\bar{c}_{ij}$ and $\bar{t}_{ij}$ the nominal fixed and variable costs from i to j, and by $\tilde{c}_{ij}$ and $\tilde{t}_{ij}$ their maximal deviations. Without loss of generality, we assume that, for each connection $(i, j) \in V_1 \quad V_2$, the actual realizations for the costs are determined by the same variable ξ_{ij}. In other words, we have $c_{ij} = \bar{c}_{ij} + \xi_{ij}\tilde{c}_{ij}$ and $t_{ij} = \bar{t}_{ij} + \xi_{ij}\tilde{t}_{ij}$, with $\boldsymbol{\xi} \in \Xi$ and Ξ is a given uncertainty set (see Section 3.4.2).

We consider the adjustable robust version of this uncertain problem, where capacity installation is determined at the first stage whereas product flows are determined after uncertainty reveals. We denote the resulting problem as ARCCFLP (for Adjustable Robust Congested Capacitated Facility Location Problem).

3.4.2 Mathematical formulation

To model ARCCFLP, we introduce, for each site $i \in V_1$, first-stage variables x_i and q_i; the former takes the value 1 if site i is activated, whereas the latter denotes the actual capacity installed. The feasible set X for the first-stage variables is defined as

$$X = \{(\boldsymbol{x}, \boldsymbol{q}) : x_i \in \{0, 1\} \text{ and } 0 \quad q_i \quad \bar{q}_i x_i \qquad \forall i \in V_1\}. \tag{3.26}$$

Once the actual realization of uncertainty $\boldsymbol{\xi} \in \Xi$ is known, thus defining the transportation costs, the remaining decisions concerning the flow of product from opened sites to clients must be taken. To this aim we introduce, for each connection $(i, j) \in V_1 \quad V_2$, variables z_{ij} and y_{ij} denoting if the connection is activated and the fraction of request of client j that is served by site i, respectively. For each site i, we also denote by v_i the total amount of product leaving the site.

Accordingly, the feasible set $Y(\boldsymbol{x}, \boldsymbol{q})$ associated with a given pair $(\boldsymbol{x}, \boldsymbol{q})$ is defined by the following constraints.

$$\sum_{j \in V_2} d_j y_{ij} \quad q_i \qquad \forall i \in V_1 \tag{3.27}$$

$$\sum_{i \in V_1} y_{ij} = 1 \qquad \forall j \in V_2 \tag{3.28}$$

$$y_{ij} \quad z_{ij} \qquad \forall i \in V_1, \forall j \in V_2 \tag{3.29}$$

$$v_i = \sum_{j \in V_2} d_j y_{ij} \qquad \forall i \in V_1 \tag{3.30}$$

$$y_{ij} \quad 0 \qquad \forall i \in V_1, \forall j \in V_2 \tag{3.31}$$

$$z_{ij} \in \{0, 1\} \qquad \forall i \in V_1, \forall j \in V_2 \tag{3.32}$$

Constraints (3.27) enforce that the total demand leaving each site does not exceed the installed capacity, while constraints (3.28) impose that, for each client, all the demand is served.

Constraints (3.29) activate connections with a positive flow. Finally, (3.30) define the total demand served by each site, whereas (3.31) and (3.32) give the domain of the variables.

Then, ARCCFLP is formulated as

$$\min_{(\boldsymbol{x}, \boldsymbol{q}) \in X} \left\{ \sum_{i \in V_1} (f_i x_i + u_i q_i) + \max_{\boldsymbol{\xi} \in \Xi} \min_{(\boldsymbol{z}, \boldsymbol{y}, \boldsymbol{v}) \in Y(\boldsymbol{x}, \boldsymbol{q})} \sum_{i \in V_1} \left(F_i(v_i) + \sum_{j \in V_2} \left((c_{ij} + \xi_{ij} \hat{c}_{ij}) z_{ij} + (t_{ij} + \xi_{ij} \hat{t}_{ij}) y_{ij} \right) \right) \right\}. \tag{3.33}$$

By applying the methodology introduced in this paper, the corresponding lower-bounding problem is given as follows

$$\min \sum_{i \in V_1} \left(f_i x_i + u_i q_i + r_i + \sum_{j \in V_2} (c_{ij} z_{ij} + t_{ij} y_{ij}) \right) \tag{3.34}$$

$$+ \max_{\boldsymbol{\xi} \in \Xi} \sum_{i \in V_1} \sum_{j \in V_2} \xi_{ij} (\hat{t}_{ij} z_{ij} + \hat{c}_{ij} y_{ij}) \tag{3.35}$$

$$\text{s.t. } (\boldsymbol{x}, \boldsymbol{q}) \in X \tag{3.36}$$

$$(\boldsymbol{q}, \boldsymbol{r}, \boldsymbol{v}, \boldsymbol{y}, \boldsymbol{z}) \in \operatorname{conv} \left(\left\{ (\boldsymbol{q}, \boldsymbol{r}, \boldsymbol{v}, \boldsymbol{y}, \boldsymbol{z}) : \begin{array}{cccc} 0 & q_i & q_i & \forall i \in V_1 \\ r_i & F_i(v_i) & & \forall i \in V_1 \\ (3.27) & (3.32) & & \end{array} \right\} \right). \tag{3.37}$$

The inner maximization problem can then be expressed by using Fenchel duality, and the resulting formulation depends on the uncertain set. In our experiments, we consider two widely used uncertainty sets, namely, the Γ-uncertainty set and the ellipsoidal uncertainty set.

Γ-uncertainty (see, Bertsimas and Sim [2004]) assumes that uncertainty affects the cost of at most Γ arcs, where Γ is a parameter used to control the robustness of the solution. Namely,

$$\Xi_\Gamma^{\triangleright} = \left\{ \boldsymbol{\xi} \in [0, 1]^{|V_1| \, |V_2|} : \sum_{i \in V_1} \sum_{j \in V_2} \xi_{ij} \quad \Gamma \right\}. \tag{3.38}$$

In this case, Fenchel duality reduces to LP duality as follows.

$$\max_{\xi \in \Xi_\Gamma} \sum_{i \in V_1} \sum_{j \in V_2} \xi_{ij}(\bar{t}_{ij} z_{ij} + \bar{c}_{ij} y_{ij}) = \min \quad \lambda + \sum_{i \in V_1} \sum_{j \in V_2} \pi_{ij} \tag{3.39}$$

$$\text{s.t. } \lambda + \pi_{ij} \quad \bar{t}_{ij} z_{ij} + \bar{c}_{ij} y_{ij} \qquad \forall i \in V_1, \forall j \in V_2 \tag{3.40}$$

$$\lambda \quad 0 \tag{3.41}$$

$$\pi_{ij} \quad 0 \qquad \forall i \in V_1, \forall j \in V_2 \tag{3.42}$$

The Ellipsoidal uncertainty set is defined as

$$\Xi_\Gamma = \left\{ \xi \in [0,1]^{|V_1| \cdot |V_2|} : \sqrt{\sum_{i \in V_1} \sum_{j \in V_2} \xi_{ij}^2} \quad \Gamma \right\} \tag{3.43}$$

where again Γ is a control parameter. In this case, one obtains the following formulation.

$$\max_{\xi \in \Xi_\Gamma} \sum_{i \in V_1} \sum_{j \in V_2} \xi_{ij}(\bar{t}_{ij} z_{ij} + \bar{c}_{ij} y_{ij}) = \min \quad \lambda + \sum_{i \in V_1} \sum_{j \in V_2} \pi_{ij} \tag{3.44}$$

$$\text{s.t. } \nu_{ij} + \pi_{ij} \quad \bar{t}_{ij} z_{ij} + \bar{c}_{ij} y_{ij} \qquad \forall i \in V_1, \forall j \in V_2 \tag{3.45}$$

$$\sqrt{\sum_{i \in V_1} \sum_{j \in V_2} \nu_{ij}^2} \quad \lambda \tag{3.46}$$

$$\lambda \quad 0 \tag{3.47}$$

$$\pi_{ij} \quad 0 \qquad \forall i \in V_1, \forall j \in V_2 \tag{3.48}$$

$$\nu_{ij} \quad 0 \qquad \forall i \in V_1, \forall j \in V_2 \tag{3.49}$$

An interested reader may refer to Li et al. [2011] for associated theoretical properties of both uncertainty sets, including their robust counterparts and probabilistic guarantees for linear constraints.

3.4.3 Test bed

Instance generation We tested our solution method on random instances, that were generated by following the guidelines of the extensive computational study by Cornuéjols et al. [1991]. Accordingly, for each facility $i \in V_1$, the maximum capacity $\bar{q}_i$ and the fixed opening cost f_i follow uniform distributions in $[10, 160]$ and $[0, 180]$, respectively, whereas the variable coefficient u_i was generated in $[200/\sqrt{\bar{q}_i}, 220/\sqrt{\bar{q}_i}]$. Moreover, locations for each facility $i \in V_1$ and each client $j \in V_2$ were generated in the unit square, and nominal transportation costs were set to the Euclidean distance multiplied by 10 and rounded up. The demands were uniformly generated between 0 and 1 and scaled so that $\sum_{i \in V_1} \bar{q}_i / \sum_{j \in V_2} d_j = \nu$ where ν is a parameter taking value in $\{1.1, 1.2, 1.3\}$. In addition, following Desrochers et al. [1995], for each $i \in V_1$, we used $\gamma_i = 1$ and $\alpha_i = \beta_i = 0.75$, i.e., each function F_i is quadratic with respect to the amount of product leaving site i. Concerning the parameters affected by uncertainty, the maximum deviation $\tilde{t}_{ij}$ was set to 0.50 $\bar{t}_{ij}$, thus allowing a maximum of 50% deviation. As to the opening cost of each arc, we randomly generated the nominal value between 50 and 100, allowing a maximum of 50% deviation with respect to this value.

Finally, the number of sites and clients take values $(4, 8)$, $(5, 10)$ and $(6, 12)$. For each combina-

tion of jV_1j, jV_2j, and ν, we generated 5 test-cases. Each test-case was solved for $\Gamma = 1, 2, 3, 4$, both in the Γ-uncertainty and in the Ellipsoidal uncertainty settings, thus producing 360 instances.

3.4.4 Implementation details

We implemented our branch-and-price algorithm with spatial branching in C++ and run all the experiments on an AMD Ryzen 5 PRO 4650GE at 3.3 GHz, with a time limit equal to 10,800 CPU seconds per run (3 hours).

At the root node, an initial upper bound is computed by solving the single-stage version of ARCCFLP where all decisions are taken here and now. This bound is obtained by solving (3.34)-(3.37) without convexifying the wait-and-see feasible space in constraint (3.37). At each node of the algorithm, we solve the restricted master problem by using Mosek 9.2, and the pricing problem by means of IBM CPLEX 22.10. This combination of solvers turned out to be the most numerically stable on our instances.

The column-generation procedure includes stabilization by dual price smoothing, as described in Pessoa et al. [2018]; and at most one column is added to the restricted master problem at each iteration.

For the branching strategy of continuous variables, we used a tolerance of $\varepsilon = 10^{-3}$ for comparing real numbers in finite precision. Local bounds derived from branching decisions are applied to the column generation sub-problem. Columns in the restricted master problem are checked against the local bounds, and possibly removed from the master.

At a given node, to check the optimality of a first-stage decision and possibly fathom the node, we use the sufficient condition from Proposition 2. If the latter does not hold, we check if all active columns at optimality are built on the same values for the continuous variables; in this case, Proposition 2 can be exploited to ensure local optimality of the corresponding solution. If the first-stage solution is not feasible, an upper bound is computed as follows. We detect the variable with largest value in the RMP current solution, recover the values of variables $(\boldsymbol{x}, \boldsymbol{q})$ that were used for generating this column and fix variables $(\boldsymbol{x}, \boldsymbol{q})$ to those values, possibly rounding up integer variables.

Finally, observe that branching may induce infeasibility in the second stage. To early detect this situation, at a given node of the branch-and-bound tree, we simply check if $\sum_{i \in V_1} x_i^u q_i^u < \sum_{j \in V_2} d_j$ holds, where x_i^u and q_i^u denote the local upper bound of variable x_i and q_i, respectively. In this case the node is declared infeasible.

3.4.5 General results

Table 3.1 reports our computational results on ARCCFLP. The upper part relates to experiments done with the Γ-uncertainty set (Γ-unc.), while the lower part addresses those with the ellipsoidal uncertainty set (Ellips.). Columns jV_1j, jV_2j and Γ give the number of sites, the number of clients and the value for the uncertainty parameter Γ, respectively. Column "solved" reports the number of instances (out of 15) which could be solved to proven optimality within the given time limit. Into brackets we report the number of instances for which the computation was stopped due to numerical issues of the used solvers. For the sake of consistency, all remaining columns but the last one are relative to instances which could be solved within the time limit. In particular,

columns "time" report, from left to right, the average time needed to prove optimality ("total"), the average time spent solving the RMP ("RMP") and the average time spent solving the pricing problem ("PP") during the execution of our Branch-And-Price algorithm. All times are expressed in seconds. Column "nodes" reports the average number of explored nodes, while "columns" gives the average number of generated columns throughout the entire execution of our algorithm. Finally, we report the average optimality gap at the root node ("root") and the average optimality gap at time limit ("end") (or when the algorithm was stopped for numerical troubles). This last figure is computed only over those instances which could not be solved to optimality.

					Time (s)					Gap (%)	
	$\|V_1\|$	$\|V_2\|$	Γ	solved	total	RMP	PP	nodes	columns	root	end
Γ-unc.	4	8	1	15	2.45	0.12	2.04	4.73	42.60	0.21	
			2	15	3.19	0.14	2.68	5.00	56.53	0.21	
			3	15	4.75	0.26	4.03	5.27	84.60	0.22	
			4	15	4.74	0.22	4.10	5.27	78.60	0.23	
	5	10	1	15	44.83	0.78	42.35	5.80	110.00	0.29	
			2	15	21.83	0.47	20.27	6.60	82.20	0.31	
			3	15	36.59	0.64	34.70	7.53	125.87	0.33	
			4	15	38.33	0.80	36.16	9.40	166.80	0.34	
	6	12	1	11	319.17	1.92	313.31	4.82	139.55	0.17	0.24
			2	13	572.80	1.64	562.48	5.46	102.69	0.19	0.30
			3	13	1117.35	2.35	1110.73	6.23	162.92	0.22	0.23
			4	11	1261.83	4.63	1249.46	6.45	239.00	0.22	0.26
Ellips.	4	8	1	13 (2)	6.64	0.35	5.70	5.00	67.92	0.22	0.11
			2	14 (1)	7.46	0.52	6.25	5.00	88.79	0.22	0.30
			3	14 (1)	12.47	0.67	10.90	5.29	121.07	0.20	0.46
			4	13 (2)	11.16	0.40	10.08	4.54	86.77	0.21	0.09
	5	10	1	12 (3)	26.96	1.27	24.59	6.00	117.58	0.26	0.44
			2	12 (3)	83.59	2.34	79.28	7.17	206.75	0.32	0.38
			3	15	82.47	3.19	76.72	8.73	278.80	0.34	
			4	15	151.89	2.17	147.62	6.33	195.27	0.31	
	6	12	1	14 (1)	324.66	3.45	315.11	5.29	169.57	0.18	0.21
			2	14 (1)	420.62	4.01	412.94	5.86	202.21	0.22	0.21
			3	13	859.62	5.66	847.91	7.00	280.08	0.24	0.10
			4	13 (2)	1574.26	10.83	1551.17	10.69	526.38	0.22	0.15

Table 3.1: Computational experiments on ARCCFLP instances. Each row refers to 15 instances.

The table shows that our method is able to solve a large fraction of the instances, namely 168 in the Γ-uncertainty setting and 162 in the Ellipsoidal uncertainty setting. In most cases, the solution time required by the algorithm is quite small and solving the RMP is very fast in practice (below 7% of the total time, on average). Indeed, the most challenging subproblem solved is the pricing problem; when increasing the size of the instances the number of columns that are needed increases and each execution of the pricing problem is more time consuming. However, the solved relaxation allows to compute a very tight approximation, as the gap between lower and upper bounds at the root that is always below 0.35%, and producing small enumeration trees, in which the number of generated nodes is always below 11. Moreover, the performance of the algorithm is satisfactory also for the instances that were not solved to proven optimality, as the average residual gap at the end of the enumeration is always quite small (below 0.5%). Finally,

observe that numerical issues arise only when uncertainty belongs to the Ellipsoidal uncertainty set, a nonlinear setting in which Mosek may encounter numerical instability on some instances.

Linearized costs

As an alternative approach for both uncertainty sets described in Section 3.4.2, we considered solving a linearized approximation of ARCCFLP obtained by replacing each function F_i ($i \in V_1$) by a piecewise linear approximation. By introducing L approximation points $\{\bar{v}_{il}\}_{l=1,\dots,L}$, function F_i is underestimated by the following one:

$$\tilde{F}_i(v_i) = \max_{l=1,\dots,L} \{F_i(\bar{v}_{il}) + F_i^0(\bar{v}_{il})(v_i - \bar{v}_{il})\}. \tag{3.50}$$

In our experiments, we chose $L = 10$ and, for all $i \in V_1$, defined the approximation points to be equally distributed in the interval $[0, \bar{q}_i]$, i.e., $\bar{v}_{il} = \bar{q}_i(l-1)/(L-1)$ for $l = 1, ..., L$.

Table 3.2 reports the results obtained by using the linearized approach. For each combination of $|V_1|$ and $|V_2|$, we give for both the exact and the linearized approaches, the number of instances solved to optimality. Moreover, for the latter we report the average and maximum percentage error introduced by the linearization, computed as $\frac{z - z^L}{z}$, where z and z^L denote the optimal values of an ARCCFLP instance and of its linearized counterpart, respectively. These figures are computed with respect to instances solved by both approaches only.

			exact	linearized		
	$\|V_1\|$	$\|V_2\|$	solved	solved	avg. err.	max. err.
Γ-unc.	4	8	60	60	0.37	0.61
	5	10	60	60	0.38	0.63
	6	12	48	42	0.39	0.56
Ellips.	4	8	54	58	0.36	0.59
	5	10	54	59	0.39	0.58
	6	12	54	59	0.40	0.56

Table 3.2: Comparison of exact and linearized approaches

The table shows that the linearized approach turns out to be harder in the Γ-uncertainty setting, while it gives some improvement when the Ellipsoidal setting is considered; this is mainly due to the reduced number of instances for which we encountered numerical troubles. However, in both settings, linearization introduces a nonneglibile error when underestimating the true cost of a solution. The average percentage error, over all instances, is 0.38% and can be as large as 0.63%.

3.5 Conclusion

In this work, we studied optimization problems where part of the cost parameters are not known at decision time, and the decision flow is modeled as a two-stage process. In particular, we addressed general problems in which all constraints (including those linking the first and the second stages) are defined by convex functions and involve mixed-integer variables. To the best of our knowledge, this is the first attempt to extend the existing literature to tackle this wide class of problems.

To this aim, we derive a relaxation of the problem which can be formulated as a convex optimization problem, and embed it within a branch-and-bound algorithm where branching occurs on integer and continuous variables. By combining enumeration and on-the-fly generation of the variables, we obtain a branch-and-price scheme, for which we prove asymptotic convergence in the general mixed-integer case and give sufficient conditions for finite convergence.

In addition to the theoretical analysis, we applied our method to an optimization problem affected by objective uncertainty arising in the logistic field, namely a variant of the congested Capacitated Facility Location problem with uncertain transportation costs. Our computational experiments showed that the proposed method is able to solve medium-size instances for this problems. In addition, we provide a comparison with a natural approach based on linearization of the congestion function, showing that this alternative solution method would give marginal improvements in terms of performances though introducing a non-negligible error in terms of cost of the provided solution.

CHAPTER 4

Application: scheduling under uncertain job failure

Francois Clautiaux, Boris Detienne, Henri Lefebvre[1]

Minimizing the weighted number of tardy jobs on one machine is a classical and intensively studied scheduling problem. In this chapter, we develop a two-stage robust approach, where exact weights are known after accepting to perform the jobs, and before sequencing them on the machine. This assumption allows diverse recourse decisions to be taken in order to better adapt one's mid-term plan.

The contribution of this chapter is twofold: first, we introduce a new scheduling problem and model it as a min-max-min optimization problem with mixed-integer recourse by extending existing models proposed for the deterministic case. Second, we take advantage of the special structure of the problem to propose two solution approaches based on results from the recent robust optimization literature: namely the K-adaptability (Hanasusanto et al. [2015]) and the convexification-based approach introduced in Arslan and Detienne [2021] and generalized in Chapter 3. We also study the additional cost of the solutions if the sequence of jobs has to be decided before the uncertainty is revealed. Computational experiments are reported to analyze the effectiveness of our approaches.

4.1 Introduction

Historically, scheduling optimization problems have been solved in a deterministic fashion assuming that every input data were perfectly known at decision time. These problems have been, and still are, extensively studied in the literature. For an overview of the broad scheduling literature, the reader may refer to Graham et al. [1979], which introduced the widely used notation for scheduling problems, and to Pinedo [2016], which covers important theoretical models and significant scheduling problems occurring in the real world.

[1] The content of this chapter has been accepted for publication at *Journal of Scheduling*.

Regarding robust scheduling approaches, Aloulou and Della Croce [2008] and Yang and Yu [2002] present different complexity results for single machine problems. They show that even simple scheduling problems become NP-hard as soon as the uncertainty set contains more than one scenario. In Bougeret et al. [2018], the authors provide approximation algorithms for the problem of minimizing the weighted and unweighted sum of completion times on a single machine where the processing time of the tasks are uncertain. Problems with stochastic breakdowns on one machine (resp. two machines) are studied in Birge et al. [1990] (resp. Allahverdi and Mittenthal [1995]). Affine decision rules are proposed for two-stage robust batch process scheduling under polyhedral uncertainty in Lappas and Gounaris [2016], based on continuous-time models oriented towards chemistry applications. In van den Akker et al. [2018], a variant of $1|\!|\sum U_j$ with uncertain processing times is studied. Given a discrete scenario-based uncertainty set, one has to determine an initial sequence of jobs that is feasible for nominal processing times. At second stage, once the scenario of actual processing times is revealed, the sequence can be adapted by rejecting some jobs. The objective is to minimize the expected cost of the repaired solution. The authors propose a dynamic programming, a branch-and-bound and a branch-and-price algorithms to solve the problem exactly.

In this chapter, we introduce a new scheduling problem as an extension of the well known $1|r_j|\sum w_j U_j$ problem where the weighted sum of tardy jobs has to be minimized. In our problem, the jobs are subject to failures, which lead to additional costs. Once the uncertain parameters are revealed (i.e. the weights of the jobs), the decision maker is allowed to take discrete recourse actions: determining the sequence of jobs, outsourcing or spending more time on the jobs to fix them. We address this problem with a robust approach.

This problem has several practical applications. Consider the following example, which arises in the astronomical field when one needs to allocate observatory time. At planning time, a number of sessions have to be reserved for observation purposes, yet, many factors which can alter the quality of the observation are not known, e.g., weather, air quality (see e.g. Garcia-Piquer et al. [2017], van Rooyen et al. [2018]). As a result, it may happen that the sessions lead to lower quality observations, which can be fixed by increasing the time allocated to it, or outsourcing it to another facility. The costs involved in such situations are typically high, therefore, a worst-case-type optimization approach is appropriate.

From an operational point of view, rescheduling jobs might be difficult or costly, and decision-makers may favour recourse solutions where the modifications are easier to handle. In this case, one may seek solutions involving minor modifications to the original plan (see e.g., Bendotti et al. [2017]). The advantages of such solutions are well understood: they reduce the operational costs, reduce the possibility of error in the process, and are generally better accepted by the operators. We therefore propose an alternative version of the scheduling problem where the sequence of jobs cannot be modified after the uncertainty is revealed.

We thus consider two hard scheduling problems with integer recourse, which were never studied before. In general, solving a robust combinatorial problem with integer recourse is a Σ_2^P-hard problem (see e.g. Claus and Simmoteit [2020]), which induces that even verifying that a first-stage solution is feasible (for any realization of the uncertain parameter) is an NP-hard problem. In deterministic optimization problem, most classical scheduling problems obviously belong to NP, so the main question is generally whether the problem belongs to P or is at least as hard as any

problem in NP. This is different in robust optimization, where the question of whether a problem belongs to the NP class is crucial from both theoretical and practical points of view. For example, if the problem does not belong to NP, the problem cannot be modelled as a Mixed-Integer Linear Program (MILP) of polynomial size (unless $NP = P$). In this chapter, we show that our specific scheduling problems belong to the subclass of robust problems identified by Arslan and Detienne [2021], and, thus, are NP-complete.

We show that the two new scheduling problems can be reformulated in such a way that recent works on robust optimization can be instantiated to reformulate this problem exactly (see Arslan and Detienne [2021] and Chapter 3) or heuristically (see Hanasusanto et al. [2015]) by deterministic (exponentially large) MILP models. In both cases, our work consists in finding non-trivial efficient reformulations that lead to models satisfying the technical conditions imposed by the two general frameworks. The computational experiments confirm that the practical difficulty of both problems is considerable: even with state-of-the art methodologies, some instances with 25 jobs remain open after one hour of computing time. We also show a surprising result: forbidding to modify the order of the jobs in the second stage increases the cost of the solution only marginally.

Among the work mentioned above, only van den Akker et al. [2018] proposes exact solving approaches. The problem that we study is different in the following ways. First, we include release dates constraints and different weights for the jobs. Second, we extend the set of possible recourse actions by adding, to the possibility of keeping or rejecting a job at the second stage, the option of repairing it at the expense of an extra processing time. Third, the uncertainty is modeled in van den Akker et al. [2018] by a finite set of discrete scenarios that affects the processing times only, while we consider a polyhedral uncertainty set defining the objective function. Both papers aim at optimizing the worst-case cost, added to average cost in van den Akker et al. [2018]. In terms of methodology, although the two works use branch-and-price algorithms, the reformulations used are totally different. The work in van den Akker et al. [2018] is based on a classical deterministic equivalent formulation, where the recourse decisions for each scenario are modeled using one set of variables and constraints, whereas the current work is based on a robust two-stage programming formulation, which is rewritten as a static robust program of very large size.

In Section 4.2, we recall some useful results on the deterministic $1|r_j|\sum w_j U_j$, which will be used in the remainder of the chapter. In Section 4.3, we formally describe a first robust version of this problem, before proposing solution methods in Section 4.4. Section 4.5 is devoted to the problem version where the order of the jobs cannot be changed at the second stage. We report our computational experiments in Section 4.6 before concluding.

4.2 Minimizing the weighted number of tardy jobs: literature review

Minimizing the weighted number of tardy jobs on a single machine, denoted $1|r_j|\sum w_j U_j$ in the literature, is a well known NP-hard scheduling problem (see Graham et al. [1979]) and can be stated as follows.

PROBLEM: $1|r_j|\sum w_jU_j$ (DECISION)
INPUT DATA: $(V, J, (r, d, w, p))$, where V is a positive value, J a set of jobs, each of which are characterized by the following data: r_j: a release date (i.e., the time before which the job cannot start); d_j: a due date (i.e., the time after which the job is considered tardy; w_j: a weight (i.e., the fixed cost for executing the job tardy) ; p_j: a processing time (i.e., the time needed to execute the job).
QUESTION: Is there a permutation σ of the tasks whose cost (i.e., the weighted number of tardy jobs) is smaller than V?

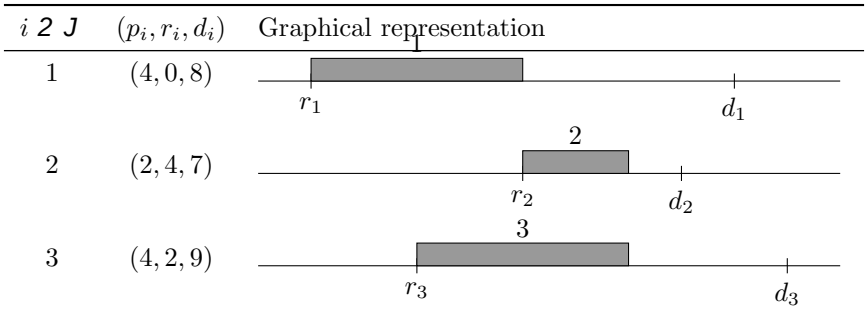
This problem has been extensively studied in the literature. In particular, Jackson [1955] proposes a dominance rule for cases with equal release dates known as the Earliest Deadline First rule. Heuristic approaches and lower bounds are given in Dauzère-Pérès and Sevaux [2003], Dauzère-Pérès [1995] while exact approaches are given in Baptiste et al. [2003], M'Hallah and Bulfin [2007], Péridy et al. [2003], Sadykov [2008]. To our knowledge, the best exact results are described in Detienne [2014], where up to 500-job instances are solved in less than one hour.

Since it is of particular interest for our approaches, we formally recall a mixed integer linear programming (MILP) formulation introduced in Detienne [2014] for solving the $1|r_j|\sum w_jU_j$ problem. The approach is based on two distinct decisions: (1) decide which jobs are to be executed tardy and (2) in what order will the on-time jobs be executed. This is possible since late jobs can be postponed arbitrarily without incurring additional costs. Moreover, we know Kise et al. [1978] that if jobs have *agreeable time windows* (i.e., the tasks can be ordered in such a way that $i < j$ implies $r_i \quad r_j$ and $d_i \quad d_j$), then a feasible sequence of on-time jobs exists if and only if the earliest due-date first rule yields a feasible solution. Therefore, an advantage of this approach is that one is able to order the jobs *a priori*, which avoids the need for variables determining the sequence of the jobs. The main idea of Detienne [2014] is to reformulate $1|r_j|\sum w_jU_j$ into the selection of jobs on a single machine, so that the dominance rule can be exploited, even if the initial instance does not have agreeable time windows. Formally, for any pair of jobs $i \in J$ and $j \in J$ such that there are solutions where j is scheduled after i and both jobs are on-time (i.e., $r_i + p_i + p_j \quad d_j$) and that have non-agreeable time windows (i.e., $r_i < r_j$ and $d_i > d_j$), a job occurrence $k \in \widetilde{J}$ is created, which represents scheduling i before j. It has a hard deadline $\bar{d}_k = d_j$, and $r_k = r_i, p_k = p_i, w_k = 0$. The original job i is also added to the set of job occurrences $\widetilde{J}$, with a null weight, and a deadline $\bar{d}_i = d_i$. The hard deadline is imposed to ensure that if the job occurrence is selected then it is scheduled on time. For every job $j \in J$, let G_j be the set gathering all job occurrences related to j. At most one job occurrence in G_j can be selected in a solution, since all of them correspond with the same original job j.

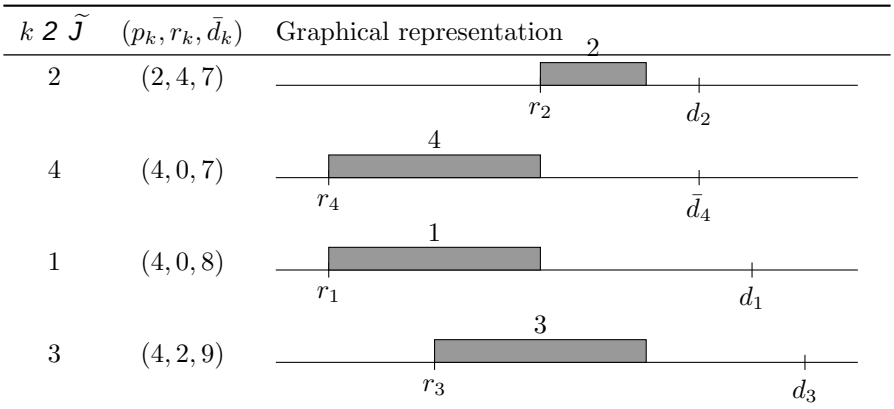
Figure 4.1 gives a small example of an initial instance defined on jobs, and a modified instance defined on job occurrences.

The following dominance rule extends the earliest deadline first rule to the general $1|r_j|\sum w_jU_j$ problem. It states that when job occurrences are built as described above, the dominance rule applies (although some pairs of time windows may not be agreeable).

Dominance rule 1 (Detienne [2014])**.** *There is at least one optimal solution to* $1|r_j|\sum w_jU_j$ *in which the selected job occurrences of the on-time jobs are ordered according to a non-decreasing order of their deadlines with ties being broken by non-decreasing order of release dates.*



Original instance



Job-occurrence instance

Figure 4.1: An instance with three jobs, and the job-occurrence representation of the instance.

Jobs 1 and 3 already have agreeable time windows, 2 and 3 have non agreeable time windows, but cannot both be scheduled on-time. Since 1 and 2 have non agreeable time windows, a job occurrence numbered 4 representing scheduling 1 before 2 is created. Occurrences are sorted according to Dominance rule 1.

In the remainder, we assume that the job occurrences are ordered according to Dominance Rule 1. Moreover, $_k$ denotes the data of the kth job occurrence in that order. For instance, p_k denotes the processing time of the kth job occurrence in that order.

We now recall in detail an MILP model, which is based on the following consideration: assume having fixed the sequencing of the on-time tasks, a straightforward way to check the feasibility of the corresponding schedule is to plan every task as soon as possible. This allows Detienne [2014] to derive an efficient MILP model for $1|r_j|\sum w_j U_j$, close to the one proposed in Dauzère-Pérès [1995]. In this model, one has to choose which jobs are late, and for each on-time job j, a job occurrence from G_j has to be selected. Then timing variables are used to ensure that job occurrences are scheduled in their hard time window. For every job $j \in J$, let U_j be a binary variable equal to 1 if j is tardy, 0 otherwise. Then, for every job occurrence $k \in G_j$, x_k is the binary variable equal to 1 if k is selected, 0 otherwise, and t_k is a variable equal to its completion time if it is scheduled, or to t_{k-1} if $x_k = 0$. The following MILP models $1|r_j|\sum w_j U_j$.

$$\min \sum_{j \in J} w_j U_j \tag{4.1}$$

$$\text{s.t.} \sum_{k \in G_j} x_k = 1 \quad U_j \quad \forall j \in J \tag{4.2}$$

$$t_k \quad \bar{d}_k \quad \forall k \in \tilde{J} \tag{4.3}$$

$$t_k \quad t_{k-1} \quad p_k x_k \quad 0 \quad \forall k > 1, k \in \tilde{J} \tag{4.4}$$

$$t_k \quad p_k x_k \quad M_k x_k \quad r_k \quad M_k \quad \forall k \in \tilde{J} \tag{4.5}$$

$$U_j \in \{0,1\} \quad \forall j \in J \tag{4.6}$$

$$x_k \in \{0,1\} \quad \forall k \in \tilde{J} \tag{4.7}$$

$$t_k \quad 0 \quad \forall k \in \tilde{J} \tag{4.8}$$

Objective function (4.1) minimizes the weighted number of tardy jobs. Constraints (4.2) enforce that exactly one job occurrence is selected for an on-time job while no job occurrence should be selected if the job is tardy.

Constraints (4.3) enforce that no job ends after its deadline, while constraints (4.4) ensure that jobs do not overlap. Finally, constraints (4.5) force each scheduled job to begin after its release date. A suitable value for constant M_k is given by $\max(0, \min_{\ell > k}\{r_\ell\})$, which ensures that if a job is not scheduled, then the value of t_k will not be larger than the smallest release date of a job occurrence of larger index.

4.3 Robust problem

In this section, we introduce a problem where the weighted number of tardy jobs has to be minimized and where the weight associated with the execution of a task is subject to uncertainty. In $1|r_j|\sum w_j U_j$, if a job is not processed on time, it can be arbitrarily delayed. So, deciding that a job will be processed late can be seen as the decision not to accept the order at all. Thus the cost associated with the late job can be seen as a loss of income.

In the two-stage robust version of $1|r_j|\sum w_j U_j$, we assume that the precise weights w_j are known only after deciding which jobs will be on time, and before processing them. Such hypothesis

holds for example when orders are accepted at a mid-term level without precise knowledge of the future technical difficulties that may arise from specific jobs, incurring extra production costs.

Our approach is also a conservative approximation of multi-stage decision processes, where the actual production costs would be revealed along time. To the best of our knowledge, uncertain multi-stage integer problems are far out of reach of existing optimization methodologies (see for example Georghiou et al. [2019] or Shapiro [2012] for algorithms dedicated to multi-stage continuous optimization problems), so that such an approximation can still be useful.

4.3.1 Problem description

We now consider that the executed jobs may fail in such a way that the output of the task is deteriorated, leading to an additional cost. When a deterioration is detected, the decision maker can take recourse decisions of three types: (1) accept the output of the job as it is, thus undertaking a penalty for producing faulty goods; (2) spend more time on the job to perform it correctly and thus avoiding the additional cost or (3) outsource the execution of jobs. Note that it is possible to outsource any job j, even if its weight w_j has not been modified, since it gives additional time for other jobs to be executed or repaired.

We assume that the maximum possible penalty for the faulty production of each job $j \in J$ is known, and denoted by $\bar{\delta}_j$. The penalty to pay for a given task $j \in J$ is computed as $\bar{\delta}_j \xi_j$, where ξ_j is the (uncertain) ratio of penalty incurred by j. Following the robust optimization framework, vector $\boldsymbol{\xi} \in \mathbb{R}^{|J|}$ belongs to the *uncertainty set* $\Xi \subseteq \mathbb{R}^{|J|}$. In this chapter, we consider the *budgeted uncertainty set*, recalled as here,

$$\Xi = \left\{ \boldsymbol{\xi} \in \mathbb{R}_+^{|J|} \middle| \xi_j \leq 1, \forall j \in J \text{ and } \sum_{j \in J} \xi_j \leq \Gamma \right\}$$

where Γ is referred to as the uncertainty budget or uncertainty parameter. This uncertainty set was introduced in Bertsimas and Sim [2004] and is conservative in the following sense: if Γ increases, the size of Ξ increases. A given vector $\boldsymbol{\xi} \in \Xi$ can be interpreted as a job failure scenario. In any scenario, at most Γ jobs can incur their maximum penalty, but more of them can be partially impacted since vector x_i does not have to take integer values, and the worst-case scenario is generally not an extreme point of Ξ in the two-stage robust optimization context. Remark that when $\Gamma = |J|$, the uncertainty set embeds every possible job failure scenarios.

The decision flow goes as follows: in a here-and-now phase (or first stage), the decision maker decides a set of jobs to be executed on time, then, as the jobs fail, the decision maker is allowed, in a wait-and-see phase (or second stage), to take recourse actions. Note that the optimal solution may be to decide to execute more tasks on time than what is feasible and to finally outsource some of these jobs to restore feasibility.

The problem can now be enunciated formally as follows. We first describe the (second-stage) repairing problem.

PROBLEM: REPAIRING PROBLEM (DECISION)
INPUT DATA: $(V, J, (r, d, w, p, \bar{\delta}, \tau, f), A, \xi)$, where $V \in \mathbb{R}$ is a target value, J, a set of jobs characterized by data (r, d, w, p), and for each job j, a maximum additional cost $\bar{\delta}_j$ if j fails, a fixed extra time τ_j needed to repair j, a fixed cost f_j for outsourcing j, a set of initially on-time jobs A, and $\bar{\xi}$ is a failure scenario.
QUESTION: Is there a partition (B, C, D) of A and a permutation σ of $B \cup C$, where B is the set of jobs to be scheduled without modification, C is the set of jobs to be fixed and scheduled, and D is the set of jobs to be outsourced, such that all jobs of $B \cup C$ are on-time and $\sum_{j \in J \setminus A} w_j + \sum_{j \in B} \bar{\delta}_j \xi_j + \sum_{j \in D} f_j \quad V$?

Using this definition, one can enunciate the two-stage problem formally.

PROBLEM: ROBUST $1|r_j|\sum w_j U_j$ (DECISION)
INPUT DATA: $(V, J, (r, d, w, p, \bar{\delta}, \tau, f), \Xi)$, where $V \in \mathbb{R}$ is a target value, J, a set of jobs characterized by data (r, d, w, p), and for each job j, a maximum additional cost $\bar{\delta}_j$ if j fails, a fixed extra time τ_j needed to fix j, a fixed cost f_j for outsourcing j, and Ξ is an uncertainty set.
QUESTION: Is there a subset $A \quad J$ of on-time jobs such that for any scenario $\xi \in \Xi$, REPAIRING PROBLEM with data $(V, J, (r, d, w, p, \bar{\delta}, \tau, f), A, \xi)$ has answer yes?

Our scheduling problem consists in seeking the minimum value of V such that the decision problem has answer yes. This optimization problem will be further referred to as problem (P). When $\bar{\delta}_j = 0$ for any job j, the problem becomes the classical $1|r_j|\sum w_j U_j$ and is therefore NP-hard. However, note that for given input data, if one is given a subset A of jobs, determining if the repairing problem has solution yes for all possible scenarios is not straightforward. This means that in terms of theoretical complexity, a first-stage solution A does not provide a direct polynomial certificate, as would be the case in a deterministic setting. We need to introduce complex reformulations before being able to prove that the problem belongs to class NP, and so, is NP-complete.

4.3.2 Formulation

In this section, we show that (4.1)-(4.8) can be extended to model problem (P). To do so, we use an approach similar to Detienne [2014] and which was recalled in Section 4.2. Its validity is based on dominance rule 1, we therefore need to ensure that it still holds for the robust case.

We create a set of job occurrences $\widetilde{J}$ from the original set of jobs J in order to turn the problem into a job occurrence selection problem with agreeable time windows. Formally, consider a job $i \in J$, for any job $j \in J$ whose time window is included in that of i (i.e., $r_i < r_j$, $d_i > d_j$), and such that i and j can both be on-time in a solution (i.e. $r_i + p_i + p_j \quad d_j$), we create a job occurrence $k \in \widetilde{J}$ such that $r_k = r_i, p_k = p_i, w_k = 0, f_k = f_i, \bar{\delta}_k = \bar{\delta}_i, \tau_k = \tau_i$ and $d_k = d_j$. Again, the original job i is also added to $\widetilde{J}$ and we introduce the set G_j as the set of job occurrences related to a given job j.

We can now extend the dominance rule 1 in the following sense:

Dominance rule 2. *There is at least one optimal solution* $((B\ ,C\ ,D\ ),\sigma\ )$ *for REPAIRING PROBLEM such that jobs of* $B\ \ [\ C$ *are scheduled according to a non-decreasing order of their deadlines*

Proof. Let $((B,C,D),\sigma)$ be a feasible solution for REPAIRING PROBLEM, and consider two jobs i and j such that $i,j\ 2\ B\ [\ C$. Assume moreover that i is before j in σ.

If $d_i < d_j$ (or $d_i = d_j$ and $r_i < r_j$), then i and j are already scheduled in the desired order. Otherwise, for any job ℓ, let $\hat{p}_\ell = p_\ell$ if $\ell\ 2\ B$ and $\hat{p}_\ell = p_\ell + \tau_\ell$ if $\ell\ 2\ C$. Note that since (B,C,D) is feasible, it holds that

$$r_i + \hat{p}_i + \hat{p}_j \quad d_j. \tag{4.9}$$

Two cases remain:

- $d_i > d_j$ and $r_i < r_j$: (4.9) implies that $r_i + p_i + p_j \quad d_j$. Therefore, there exists a job occurrence $k\ 2\ \mathsf{G}_i$ such that $d_k = d_j$ and with which we can replace i. Doing so, we end up in the desired order and the objective value remains unchanged.
- $d_i \quad d_j$ and $r_i \quad r_j$: swapping i and j leads to the desired order, without modifying the cost of the solution. Since $r_j \quad r_i$ and $d_i \quad d_j$, it holds from (4.9) that $r_j + \hat{p}_j + \hat{p}_i \quad d_i$, which shows that the solution remains feasible. □

□

In the exact same way as what has been done in Detienne [2014] and recalled in Section 4.2, we sort the job occurrences according to a non-decreasing order of their deadlines.

We now propose a characterization of valid recourse decisions. Schedule-feasibility of a selection of jobs has been studied in the static case by Detienne [2014] (see Section 4.2) and can be extended to our case.

For any $k\ 2\ \widetilde{J}$, binary variable y_k takes value 1 if k is selected, 0 otherwise ; variable z_k takes value 1 if k is repaired, 0 otherwise. For each occurrence $k\ 2\ \widetilde{J}$, variable $\rho_k\ 2\ \mathsf{R}_+$ is equal to the processing time of the job (including possible repairing).

We define the set $\bar{Y} \quad \mathsf{f}0,1\mathsf{g}^{2j\widetilde{J}j} \quad \mathsf{R}_+^{2j\widetilde{J}j}$, which contains every feasible schedule, as follows.

$$\bar{Y} = \begin{cases} \rho_k = p_k y_k + \tau_k z_k \quad 8k\ 2\ \widetilde{J} & (4.10)\\ z_k \quad y_k \quad 8k\ 2\ \widetilde{J} & (4.11)\\ \sum\limits_{k2\mathsf{G}_j} y_k \quad 1 \quad 8j\ 2\ J & (4.12)\\ t_k \quad \bar{d}_k \quad 8k\ 2\ \widetilde{J} & (4.13)\\ t_k \quad t_{k\ 1} \quad \rho_k \quad 0 \quad 8k>1, k\ 2\ \widetilde{J} & (4.14)\\ t_k \quad \rho_k \quad M_k y_k \quad r_k \quad M_k \quad 8k\ 2\ \widetilde{J} & (4.15)\\ t_k \quad 0 \quad 8k\ 2\ \widetilde{J} & (4.16)\\ y_k, z_k\ 2\ \mathsf{f}0,1\mathsf{g} \quad 8k\ 2\ \widetilde{J} & (4.17)\\ \rho_k \quad 0 \quad 8k\ 2\ \widetilde{J} & (4.18) \end{cases}$$

Constraints (4.10) ensure that each value ρ_k is equal to the processing time of k with respect to the recourse action. Constraints (4.11) enforce that a job may be fixed only if it is scheduled while constraints (4.12)-(4.17) are understood exactly as (4.1)-(4.8) where the constant processing times p have been substituted by decision variables ρ. Recall that a job can be outsourced, which explains why constraints (4.12) are less-or-equal constraints whereas (4.2) are equality constraints.

Let us also denote the set of second-stage decisions that admit a feasible timing of the jobs by

$$Y = \{(\boldsymbol{y}, \boldsymbol{z}) \mid \exists \boldsymbol{t}, \boldsymbol{\rho} : (\boldsymbol{y}, \boldsymbol{z}, \boldsymbol{t}, \boldsymbol{\rho}) \in \bar{Y}\}.$$

To enforce the non-anticipation property, which stipulates that the decided recourse action may not contradict the first-stage decision, we add so-called linking constraints between the first-stage decisions and the second-stage decisions. These linking constraints are expressed as

$$\sum_{k \in G_j} y_k \quad 1 \quad U_j \quad \forall j \in J \tag{4.19}$$

We also introduce the set $Y(\boldsymbol{U})$ of admissible recourse decisions respecting both the non-anticipation and the schedule-feasibility property. It is given by $Y(\boldsymbol{U}) = \{(\boldsymbol{y}, \boldsymbol{z}) \in Y \mid (4.19)\}$.

We now take interest in the objective value. Let $j \in J$ be a job to be scheduled, then:

- if $U_j = 1$, then j is executed tardy and we have $\forall k \in G_j, y_k = z_k = 0$ (i.e., no recourse action)
- if $U_j = 0$, then j is accepted in the first stage. In the second stage (i.e., once the uncertainty is revealed) the sequencing of the jobs has to be decided as well as the recourse actions. The following cases may arise:
  - there is $k \in G_j$ such that $y_k = z_k = 1$: the job is executed and fixed
  - there is $k \in G_j$ such that $y_k = 1$ and $z_k = 0$: the job is executed
  - for all $k \in G_j, y_k = z_k = 0$: the job is outsourced.

The problem can finally be cast as:

$$(P) : \min_{\boldsymbol{U} \in \{0,1\}^{|J|}} \sum_{j \in J} w_j U_j + f_j(1 \quad U_j) + \max_{\boldsymbol{\xi} \in \Xi} \min_{(\boldsymbol{y}, \boldsymbol{z}) \in Y(\boldsymbol{U})} R(\xi, y, z)$$

where $R(\boldsymbol{\xi}, \boldsymbol{y}, \boldsymbol{z})$ denotes the cost of recourse action $(\boldsymbol{y}, \boldsymbol{z})$ corresponding to scenario ξ given by:

$$R(\boldsymbol{\xi}, \boldsymbol{y}, \boldsymbol{z}) = \sum_{j \in J} \sum_{k \in G_j} \left[(\bar{\delta}_k \xi_j \quad f_k) y_k \quad \bar{\delta}_k \xi_j z_k \right].$$

Note that the outsourcing cost appears both in the first-stage and second-stage objective functions. This technical manipulation is required to preserve their linearity, and can be interpreted as always paying outsourcing, unless the job is scheduled in the second stage.

4.4 Solution approaches

In this section, we develop two solution approaches for solving this min-max-min problem based on two recent studies on robust optimization. First, we present the K-adaptability approach, which uses restrictive assumptions on the recourse set. Then, an exact approach is developed based on polyhedral results introduced in Arslan and Detienne [2021], which are further briefly recalled for the sake of completeness.

4.4.1 K-Adaptability

The derivation of an exact MILP model of the finite adaptability approximation for robust problems with recourse where the uncertainty is confined in the objective function has been summarized in Hanasusanto et al. [2015] and is well established (see also Chapter 2). In our setting, this consists in deciding at the first stage, in addition to the set of jobs accepted (*i.e.* a vector $\boldsymbol{U}$), K recourse solutions $(\boldsymbol{y}^q, \boldsymbol{z}^q) \in Y(\boldsymbol{U})$, $q = 1, \ldots, K$, each of which prescribing which jobs should be executed and repaired if we select this specific recourse solution. The second stage reduces to choosing one of these recourse solutions, once the uncertain parameters $\boldsymbol{\xi}$ are revealed. We get the following model:

$$\min \sum_{j \in J} w_j U_j + f_j(1 \quad U_j) + \max_{\boldsymbol{\xi} \in \Xi} \min_{q=1,\ldots,K} R(\boldsymbol{\xi}, \boldsymbol{y}^q, \boldsymbol{z}^q) \tag{4.20}$$

$$\text{s.t. } (\boldsymbol{y}^q, \boldsymbol{z}^q) \in Y(\boldsymbol{U}) \quad q = 1, \ldots, K \tag{4.21}$$

$$\boldsymbol{U} \in \{0,1\}^{|J|} \tag{4.22}$$

The reformulation process proposed in Hanasusanto et al. [2015] goes by writing the max-min problem in (4.20) as a single stage maximization linear program (LP), by making use of an epigraph formulation of the inner finite minimum. Using LP duality, an equivalent minimization LP model is derived for expressing the cost of the second-stage max-min sub-problem. Integrated into the first-stage model, this yields a bilinear model which is further linerarized with help of additional decision variables.

More precisely, let us focus on the inner maximization problem in (4.20) and employ an epigraph formulation of the finite minimum, we get that
$\max_{\boldsymbol{\xi} \in \Xi} \min_{q=1,\ldots,K} R(\boldsymbol{\xi}, \boldsymbol{y}^q, \boldsymbol{z}^q)$ is equivalent to the following problem:

$$\max \theta \tag{4.23}$$

$$\text{s.t. } \theta \quad R(\boldsymbol{\xi}, \boldsymbol{y}^q, \boldsymbol{z}^q) \quad q = 1, \ldots, K \qquad (\beta_q \quad 0) \tag{4.24}$$

$$\sum_{j \in J} \xi_j \quad \Gamma \qquad (u \quad 0) \tag{4.25}$$

$$\xi_j \quad 1 \quad \forall j \in J \qquad (v_j \quad 0) \tag{4.26}$$

$$\xi_j \quad 0 \quad \forall j \in J \tag{4.27}$$

$$\theta \in \mathbb{R} \tag{4.28}$$

where constraints (4.25)-(4.27) ensure that $\boldsymbol{\xi} \in \Xi$. Observe that model (4.23)-(4.28) is a feasible

and bounded linear program. We can then use the strong duality theorem in linear programming to obtain an equivalent dual linear program, where $\boldsymbol{\beta}$, u and $\boldsymbol{v}$ are the vectors of dual variables respectively associated with constraints (4.24), (4.25) and (4.26) of conforming dimensions. For the sake of completeness, we note that the fully developed expression of Constraints (4.24) at rank q is:

$$\theta \quad \sum_{j \in J} \left[\sum_{k \in G_j} \bar{\delta}_k (z_k^q \quad y_k^q) \right] \xi_j \quad \sum_{j \in J} \sum_{k \in G_j} \quad f_k y_k^q.$$

The dual program reads:

$$\min \quad \sum_{q=1}^{K} \sum_{j \in J} \sum_{k \in G_j} f_k y_k^q \beta_q + \Gamma u + \sum_{j \in J} v_j \tag{4.29}$$

$$\text{s.t.} \sum_{q=1}^{K} \beta_q = 1 \qquad (\theta \in \mathbb{R}) \tag{4.30}$$

$$\sum_{k \in G_j} \sum_{q=1}^{K} \bar{\delta}_k \, (z_k^q \quad y_k^q) \, \beta_q + u + v_j \quad 0 \quad \forall j \in J \qquad (\xi \quad 0) \tag{4.31}$$

$$\beta_q \quad 0, q = 1, \ldots, K \tag{4.32}$$

$$v_j \quad 0, \forall j \in J \tag{4.33}$$

$$u \quad 0 \tag{4.34}$$

This equivalent formulation contains a bilinear term in $(\boldsymbol{y}, \boldsymbol{z})$ and $\boldsymbol{\beta}$, which can be linearized using standard techniques and introducing auxiliary variables such that $\psi_k^q = y_k^q \beta_q$ and $\zeta_k^q = z_k^q \beta_q$ for all $q = 1, \ldots, K$ and $k \in \widetilde{J}$. Doing so, we obtain the following MILP finite adaptability formulation:

$$\min \sum_{j \in J} [w_j U_j + f_j (1 \quad U_j) + v_j] + \Gamma u \quad \sum_{q=1}^{K} \sum_{k \in \widetilde{J}} f_k \psi_k^q \tag{4.35}$$

$$\text{s.t.} \; \rho_k^q = p_k y_k^q + \tau_k z_k^q \quad \forall k \in \widetilde{J}, q = 1, \ldots, K \tag{4.36}$$

$$t_k^q \quad \bar{d}_k \quad \forall k \in \widetilde{J}, q = 1, \ldots, K \tag{4.37}$$

$$t_k^q \quad t_{k \; 1}^q \quad \rho_k^q \quad 0 \quad \forall k > 1, k \in \widetilde{J}, q = 1, \ldots, K \tag{4.38}$$

$$t_k^q \quad \rho_k^q \quad M_k y_k^q \quad r_k \quad M_k \quad \forall k \in \widetilde{J}, q = 1, \ldots, K \tag{4.39}$$

$$z_k^q \quad y_k^q \quad \forall k \in \widetilde{J}, q = 1, \ldots, K \tag{4.40}$$

$$\sum_{k \in G_j} y_k^q \quad 1 \quad U_j \quad \forall j \in J, q = 1, \ldots, K \tag{4.41}$$

$$\sum_{k \in G_j} \sum_{q=1}^{K} \bar{\delta}_k \, (\zeta_k^q \quad \psi_k^q) + u + v_j \quad 0 \quad \forall j \in J \tag{4.42}$$

$$\psi_k^q \quad y_k^q \quad \forall k \in \widetilde{J}, q = 1, \ldots, K \tag{4.43}$$

$$\psi_k^q \quad \beta_q \quad 8k \ 2 \ \widetilde{J}, q = 1, \ldots, K \tag{4.44}$$

$$\psi_k^q \quad \beta_q \quad 1 + y_k^q \quad 8k \ 2 \ \widetilde{J}, q = 1, \ldots, K \tag{4.45}$$

$$\zeta_k^q \quad z_k^q \quad 8k \ 2 \ \widetilde{J}, q = 1, \ldots, K \tag{4.46}$$

$$\zeta_k^q \quad \beta_q \quad 8k \ 2 \ \widetilde{J}, q = 1, \ldots, K \tag{4.47}$$

$$\zeta_k^q \quad \beta_q \quad 1 + z_k^q \quad 8k \ 2 \ \widetilde{J}, q = 1, \ldots, K \tag{4.48}$$

(4.30) (4.34)

$$t_k^q \quad 0 \quad 8k \ 2 \ \widetilde{J}, q = 1, \ldots, K \tag{4.49}$$

$$U_j \ 2 \ f0, 1g \quad 8j \ 2 \ J \tag{4.50}$$

$$y_k^q \ 2 \ f0, 1g \quad 8k \ 2 \ \widetilde{J}, q = 1, \ldots, K \tag{4.51}$$

$$z_k^q \ 2 \ f0, 1g \quad 8k \ 2 \ \widetilde{J}, q = 1, \ldots, K \tag{4.52}$$

$$\psi_k^q \quad 0 \quad 8k \ 2 \ \widetilde{J}, q = 1, \ldots, K \tag{4.53}$$

$$\zeta_k^q \quad 0 \quad 8k \ 2 \ \widetilde{J}, q = 1, \ldots, K \tag{4.54}$$

Here, equation (4.35) defines the objective function to be minimized. Constraints (4.36),(4.37)-(4.39) correspond to scheduling constraints (4.10),(4.13)-(4.15) for each $q = 1, \ldots, K$, derived from Detienne [2014], enforcing that the final decision must yield a feasible schedule. Constraints (4.40) make sure that a job can be fixed only if it is scheduled while constraints (4.41) enforce that one may only process a job which was chosen to be on-time in the first stage. Constraints (4.42) correspond to the dualized cost corresponding to the uncertain event, obtained from (4.31) through linearization of the bilinear terms. Finally, constraints (4.43)-(4.48) correspond to the linearization of $y_k^q \beta_q$ and $z_k^q \beta_q$.

This problem will be referred to as problem (P_K) and its model as model KAdapt1-a. Additionally, Subramanyam et al. [2019] has introduced a generic branch-and-bound algorithm in order to solve finite adaptability approaches. Their approach is based on disjunctive programming considerations and scenario generation. We will denote this approach by KAdapt1-b.

4.4.2 Convexi cation of the recourse set

In this section, we briefly show how one can apply the results from Arslan and Detienne [2021] which were generalized in Chapter 3. In problem (P), the inner max-min problem involves continuous decision variables for the max part, and mixed-binary decision variables for the min part. Thanks to the special structure of the linking constraints (4.19), one can use Proposition 4 recalled below and employ the following reformulation process. First, replace the feasible space of the inner min sub-problem with its convex hull **(step 1)**. It is then possible to swap the max and min operators **(step 2)**. This second step leads to a static robust MILP model. The third step applies the classical LP duality-based reformulation Bertsimas and Sim [2004] to obtain a single-stage deterministic model **(step 3)**. To write a proper MILP model, the final step expresses $Y(\boldsymbol{U})$ in terms of its extreme points **(step 4)**. This step implies an exponential growth of the model, which, at solution time, is taken care of with help of a column generation algorithm.

We first detail the reformulation process applied to problem (P), and then show how the large-scale MILP model obtained can be solved.

Reformulation

To perform **step 1**, observe that the recourse cost function R is affine in $(\boldsymbol{y}, \boldsymbol{z})$. It follows that minimizing R over $Y(\boldsymbol{U})$ and $\text{conv}\,(Y(\boldsymbol{U}))$ is equivalent. Problem (P) is then equivalent to:

$$\min_{\boldsymbol{U} \in \{0,1\}^{|J|\times|J|}} \sum_{j \in J} [w_j U_j + f_j(1 \quad U_j)] + \max_{\boldsymbol{\xi} \in \Xi} \min_{(\boldsymbol{y},\boldsymbol{z}) \in \text{conv}(Y(U))} R(\boldsymbol{\xi}, \boldsymbol{y}, \boldsymbol{z})$$

Step 2: Since function R is affine in both $(\boldsymbol{y}, \boldsymbol{z})$ and $\boldsymbol{\xi}$, it is convex in $(\boldsymbol{y}, \boldsymbol{z})$ and concave in ξ. Moreover, thanks to step 1, both max and inner min operators are performed over compact convex sets, so that we can use the well-known minimax theorem Neumann [1928] to swap them. Grouping both min operators yields:

$$\min_{\substack{\boldsymbol{U} \in \{0,1\}^{|J|\times|J|} \\ (\boldsymbol{y},\boldsymbol{z}) \in \text{conv}(Y(\boldsymbol{U}))}} \left[\sum_{j \in J} [w_j U_j + f_j(1 \quad U_j)] \; + \max \left\{ R(\boldsymbol{\xi}, \boldsymbol{y}, \boldsymbol{z}) : \begin{array}{lll} \sum_{j \in J} \xi_j & \Gamma & (u \quad 0) \\ \xi_j & 1, \forall j \in J & (v_j \quad 0) \\ \xi_j & 0, \forall j \in J & \end{array} \right\} \right]$$

Step 3 relies on the fact that the inner maximization problem is a feasible and bounded LP. Using the strong LP duality theorem, one can replace it with its dual problem to get the following formulation, where u and $\boldsymbol{v}$ are dual variables associated with the constraints imposing $\boldsymbol{\xi} \in \Xi$:

$$\begin{aligned} \min \; & \sum_{j \in J} [w_j U_j + f_j(1 \quad U_j) + v_j] + \Gamma u \quad \sum_{k \in \widetilde{J}} f_k y_k \\ \text{s.t. } & (y, z) \in \text{conv}\,(Y(U)) \\ & u + v_j \quad \sum_{k \in G_j} \bar{\delta}_k \,(y_k \quad z_k) \quad \forall j \in J \\ & U_j \in \{0,1\} \quad \forall j \in J \\ & y_k, z_k \quad 0 \quad \forall k \in \widetilde{J} \\ & v_j \quad 0 \quad \forall j \in J \\ & u \quad 0 \end{aligned} \tag{4.55}$$

This model is linear except for constraint (4.55). In order to write a linear system for these conditions, **step 4** alleviates a key obstacle: considering a fixed vector $\bar{\boldsymbol{U}}$, it is easy to express the set $\text{conv}\left(Y(\bar{\boldsymbol{U}})\right)$ in terms of the extreme points of $Y(\bar{\boldsymbol{U}})$ since it is a bounded mixed-integer set. However, the set of extreme points to consider depends on the value of $\bar{\boldsymbol{U}}$. In a general setting, this naturally leads to a disjunctive formulation whose numerical solution seems to be very challenging (the reader may refer to Arslan and Detienne [2021] and Chapter 3 for details about this technical difficulty and approaches to cope with it). Problem (P) enjoys a convenient structure that allows us to use the convex hull of Y instead of $Y(\boldsymbol{U})$, and impose the restrictions over $\boldsymbol{y}$, $\boldsymbol{z}$ and $\boldsymbol{U}$ independently. That means that a single set of extreme points, independent of $\boldsymbol{U}$, can be considered in the model. To this end, we use the following key result:

Proposition 4 (Arslan and Detienne [2021])**.**
Consider the following two-stage robust mixed-integer linear problem with objective uncertainty:

$$\min_{x \in X} \left\{ c^T x + \max_{\xi \in \Xi} \min_{y \in \tilde{Y}(x)} \xi^T Q x \right\}$$

where $X \quad \{0,1\}^N \quad \mathbb{R}^M$ *denotes the set of feasible first-stage decision,* Ξ *represents the uncertainty polyhedron and* $\tilde{Y}(x)$ *denotes the set of eligible second-stage decisions defined as* $\{y \in Y | y_1 \quad x_1\}$ *with* $Y \quad \{0,1\}^N \quad \mathbb{R}^M$ *and* $y_1 \in \{0,1\}^N$, $x_1 \in \{0,1\}^N$. *It holds* $\operatorname{conv}\left(\tilde{Y}(x)\right) = \operatorname{conv}(Y) \setminus \{y | y_1 \quad x_1\}$ *for any* $x \in X$.

From which we easily derive the following corollary:

Corrolary 3.

$$\operatorname{conv}(Y(\boldsymbol{U})) = \operatorname{conv}(Y) \setminus \left\{ \begin{array}{l} y \in \mathbb{R}^{|\tilde{J}|} \\ z \in \mathbb{R}^{|\tilde{J}|} \\ t \in \mathbb{R}^{|\tilde{J}|}_+ \end{array} \middle| \sum_{k \in G_j} y_k \quad 1 \quad U_j, \forall j \in J \right\}$$

Let us denote the set of extreme points of Y by $(\mathbf{y}^e, \mathbf{z}^e), e \in E$ (E being a list for their indices). Problem (P) is finally modeled by this deterministic equivalent program:

$$[DEP] : \min F(\boldsymbol{U}, u, \boldsymbol{v}, \boldsymbol{\alpha}) = \sum_{j \in J} [w_j U_j + f_j(1 \quad U_j) + v_j] + \Gamma u \quad \sum_{k \in \tilde{J}} \left[f_k \sum_{e \in E^R} \mathbf{y}_k^e \alpha_e \right] \tag{4.56}$$

$$\text{s.t.} \sum_{e \in E} \alpha_e = 1 \tag{4.57}$$

$$\sum_{k \in G_j} \sum_{e \in E} \mathbf{y}_k^e \alpha_e \quad 1 \quad U_j \quad \forall j \in J \tag{4.58}$$

$$u + v_j \quad \sum_{k \in G_j} \left[\bar{\delta}_k \sum_{e \in E} (\mathbf{y}_k^e \quad \mathbf{z}_k^e) \alpha_e \right] \quad \forall j \in J \tag{4.59}$$

$$U_j \in \{0,1\} \quad \forall j \in J$$

$$\alpha_e \quad 0 \quad \forall e \in E$$

$$u \quad 0$$

$$v_j \quad 0 \quad \forall j \in J$$

Here, decision vector $\boldsymbol{\alpha}$ represents the convex combination multipliers from the reformulation of $\operatorname{conv}(Y)$. Again, u and $\boldsymbol{v}$ are the dual variables associated to the constraint $\xi \in \Xi$. Constraints (4.58) link the recourse action with the first-stage decision. Constraint (4.57) enforces that the recourse actions are convex combinations of the extreme points of $\operatorname{conv}(Y)$. Finally, constraints (4.59) embed the dualized cost associated to the worst case scenario. This model will be referred to as ColGen1.

We remark that such a characterization of the convex hull as its Minkowski-Weyl formulation is akin to the Dantzig-Wolfe decomposition. Unlike the typical application of Dantzig-Wolfe decomposition, there is no integrality requirements over the reformulated variables (here, the second-stage variables $\boldsymbol{y}$ and $\boldsymbol{z}$). This stems from the reformulation process that we use. Although (P) involves integer variables in the second-stage, **step 1** allows considering $\operatorname{conv}(Y(\boldsymbol{U}))$ instead

of $Y(\boldsymbol{U})$ while keeping an equivalent problem, hence dropping the integrality requirements on variables $\boldsymbol{y}$ and $\boldsymbol{z}$. It follows that the Dantzig-Wolfe reformulation is applied to a subsystem that involves only continuous variables. Note that, in optimal solutions of $[DEP]$, second-stage variables are most of the time non-integer. Intuitively, several different second-stage solutions are required to prevent the adversary that maximizes the cost of the solution from increasing the value of the first-stage solution by moving slightly the uncertain parameters.

Problem (P) is trivially NP-hard, since considering null penalties yields a problem equivalent to the deterministic $1|r_i|\sum w_i U_i$. However, it is often not clear whether two-stage robust problems lie higher in the polynomial hierarchy or not (see Bertsimas et al. [2013] for example). As a by-product, this reformulation shows that problem (P) is not harder than NP-complete problems (the result is proven in a more general setting in Arslan and Detienne [2021]).

Corrolary 4. *Problem (P) is NP-complete.*

Column generation-based solution algorithm

Model $[DEP]$ has an exponential number of variables. A classical approach to solve such problems is to use the column generation algorithm to compute its linear relaxation. In this section, we formally present the master program and the pricing problem that has to be solved in this purpose. We then describe the column generation procedure. Finally, we depict the so-called branch-and-price algorithm, which is a tree search embedding the column generation routine to compute the optimal feasible solution of $[DEP]$.

The column generation procedure solves the linear relaxation of model $[DEP]$. Its basic idea is to consider only a subset E^R of the α-variables and solve optimally the so-called restricted master program (RMP) $[DEP]^R$, using for example the simplex algorithm. The linear relaxation of RMP can be stated as follows (constraints $U_j \quad 1$ are dropped since they are implied by constraints (4.62)).

$$[DEP]^R : \min F^R(\boldsymbol{U}, u, \boldsymbol{v}, \boldsymbol{\alpha}) = \sum_{j \in J} [w_j U_j + f_j(1 \quad U_j) + v_j] + \Gamma u \quad \sum_{k \in \tilde{J}} \left[f_k \sum_{e \in E^R} \mathbf{y}_k^e \alpha_e \right] \tag{4.60}$$

$$\text{s.t.} \sum_{e \in E^R} \alpha_e = 1 \tag{4.61}$$

$$\sum_{e \in E^R} \mathbf{y}_k^e \alpha_e + U_j \quad 1 \quad \forall k \in G_j, \forall j \in J \tag{4.62}$$

$$u + v_j \quad \sum_{k \in G_j} \left[\bar{\delta}_k \sum_{e \in E^R} (\mathbf{y}_k^e \quad \mathbf{z}_k^e)\, \alpha_e \right] \quad \forall j \in J \tag{4.63}$$

$$\alpha_e \quad 0 \quad \forall e \in E^R \tag{4.64}$$

$$U_j \quad 0, v_j \quad 0 \quad \forall j \in J \tag{4.65}$$

$$u \quad 0 \tag{4.66}$$

Basic LP theory tells us that the solution obtained is optimal for the linear relaxation of $[DEP]$ if the reduced costs of all the α-variables are non-negative. Let λ, $\boldsymbol{\mu}$ and $\boldsymbol{\pi}$ be the dual variables respectively associated with constraints (4.61), (4.62) and (4.63). Given an optimal dual

solution $(\lambda\ ,\mu\ ,\pi\ )$ to $[DEP]^R$, the so-called pricing problem that seeks a minimum reduced cost α-variable can be cast as:

$$[Pricing(\lambda\ ,\boldsymbol{\mu}\ ,\boldsymbol{\pi}\ )]\ :\min G(\lambda\ ,\boldsymbol{\mu}\ ,\boldsymbol{\pi}\ ,y,z) = \quad \lambda\ + \sum_{j\,2\,J}\ \sum_{k2G_j} \left[(\quad f_k \quad \mu_k + \bar{\delta}_k\pi_j)y_k \quad \bar{\delta}_k\pi_j z_k\right]$$

$$\text{s.t. } (\boldsymbol{y},\boldsymbol{z},\boldsymbol{t},\boldsymbol{\rho})\ 2\ \bar{Y}$$

This problem can be interpreted as a variant of $1|r_i|\sum w_iU_i$ where each job comes in two possible modes (related with variables y or z), having different processing times and weights.

When the optimal solution $(\boldsymbol{U}\ ,\boldsymbol{u}\ ,\boldsymbol{v}\ ,\boldsymbol{\alpha}\ )$ of the linear relaxation of $[DEP]$ satisfies the integrality requirements (*i.e.* $\boldsymbol{U}\ \ 2\ f0,1g^{|J|}$), then it provides an optimal first-stage solution for (P). Otherwise, one has to branch in order to exclude the current fractional solution and explore the feasibility set. Algorithm 1 summarizes the branch-and-price procedure proposed to solve problem (P) through its formulation $[DEP]$. Line 1 initializes the set of columns so that the restricted master problem is feasible. The best primal bound found, $PrimalBound$ and the best feasible solution found, S are initialized in Line 2. Each node is encoded as the set of branching constraints, B, defining the set of solutions of that node. The list of open nodes, Q, is thus initialized in Line 2 with the root node, that has no branching constraints. Loop 3-14 processes the open nodes. The solution of the relaxation at the current node is computed in Line 5. If the solution satisfies the integrality requirements (Line 10), $PrimalBound$ and S are updated (line 11). When $\boldsymbol{U}$ is not integer, branching is performed in Lines 13 and 14.

Algorithm 1: Branch-and-price algorithm for solving model $[DEP]$.

```
1  Initialize the set of columns so that [DEP]^R is feasible:  (ȳ^1, z̄^1)   f(0,0)g, E^R   f1g
2  PrimalBound   1 , S   ;, Q   f;g
3  while Q 6= ; do
4      Pop a node/set of branching constraints B from Q
5      (U ,u ,v ,α )   optimizeRelaxation(B, E^R)
6      DualBound   F(U ,u ,v ,α )
7      if DualBound   PrimalBound then
8          Current node is pruned by bound
9      else
10         if U  2 f0,1g^{|J|} then
11             Update PrimalBound and S  with DualBound and (U ,u ,v ,α )
12         else
13             Choose i 2 f1,...,|J|g such that U_i  2]0,1[
14             Add two nodes B_0 = B [ fU_i = 0g and B_1 = B [ fU_i = 1g to Q
15 return S , an optimal solution of [DEP]
```

Algorithm 2 depicts the column generation procedure used to compute the relaxation at each node of the search tree in Line 5 of Algorithm 1. Loop 1-8 adds new columns to the restricted master $[DEP]^R$ until no negative reduced cost column is found. Model $[DEP]^R$ is solved in Line 2, providing optimal dual variables that are used as input to the pricing problem in Line 4. Lines 6-7 add a new column to $[DEP]^R$ if the pricing problem returns a column with a negative reduced cost.

Algorithm 2: *optimizeRelaxation*($\mathcal{B}, E^R$): column generation algorithm for computing the dual bound at each node of the search tree when solving $[DEP]$.

Input: $\mathcal{B}$: set of branching constraints, E^R: set of indices of columns

repeat
  Solve $[DEP]^R$ with additional branching constraints $\mathcal{B}$
  Let $(\boldsymbol{U}, u, \boldsymbol{v}, \boldsymbol{\alpha})$ be the optimal solution and λ, $\boldsymbol{\mu}$ and $\boldsymbol{\pi}$ be the optimal dual values associated with constraints (4.61), (4.62) and (4.63)
  Solve $[Pricing(\lambda, \boldsymbol{\mu}, \boldsymbol{\pi})]$
  Let $(\boldsymbol{y}, \boldsymbol{z}, \boldsymbol{t}, \boldsymbol{\rho})$ be the optimal solution
  if $G(\lambda, \boldsymbol{\mu}, \boldsymbol{\pi}, \boldsymbol{y}, \boldsymbol{z}) < 0$ **then**
    $E^R \quad E^R \cup \{|E^R| + 1\}, (\bar{\mathbf{y}}, \bar{\mathbf{z}})^{|E^R|} \quad (\boldsymbol{y}, \boldsymbol{z})$
until $G(\lambda, \boldsymbol{\mu}, \boldsymbol{\pi}, \boldsymbol{y}, \boldsymbol{z}) \quad 0$
return $(\boldsymbol{U}, u, \boldsymbol{v}, \boldsymbol{\alpha})$

4.5 Order- xing rst stage

In this section, we study a variant of problem ($\mathcal{P}$), denoted ($\widetilde{\mathcal{P}}$), where the first-stage decisions include not only the selection of jobs to process, but also their order. In a wait-and-see phase, the recourse actions to be decided for each accepted job are: process the job (possibly with a decreased cost), outsource the job, or repair the job. In problem ($\mathcal{P}$), one can decide the actual processing order of the jobs after knowing their true weight, while in ($\widetilde{\mathcal{P}}$) the sequence of accepted jobs is decided at first stage, and can only be amended by removing some elements of the sequence in the second stage.

We first formalize this variant, characterize its relation with problem ($\mathcal{P}$) and derive MILP formulations. The same two solution approaches, namely finite adaptability and convexification can be applied. Since their application to this variant uses the same mathematical results as the first problem, we do not report them.

4.5.1 Formulation

We formally state problem ($\widetilde{\mathcal{P}}$). Similarly to problem ($\mathcal{P}$), we first define the recourse problem.

PROBLEM: FIXED-ORDER-REPAIRING PROBLEM (DECISION)
INPUT DATA: $(V, \mathcal{J}, (r, d, w, p, \bar{\delta}, \tau, f), A, \xi, \sigma)$, where $V \in \mathbb{R}$ is a target value, $\mathcal{J}$, a set of jobs characterized by data (r, d, w, p), and for each job j, a maximum additional cost $\bar{\delta}_j$ if j fails, a fixed extra time τ_j needed to repair j, a fixed cost f_j for outsourcing j, a set of initially on-time jobs A, $\bar{\xi}$ a failure scenario, and σ a permutation of the elements of A.
QUESTION: Is there a partition (B, C, D) of A, where B is the set of jobs to be scheduled without modification, C is the set of jobs to be fixed and scheduled, and D is the set of jobs to be outsourced, σ restricted to $B \cup C$ is feasible, and
$\sum_{j \in \mathcal{J} \cap A} w_j + \sum_{j \in B} \bar{\delta}_j \xi_j + \sum_{j \in D} f_j \quad V$?

The order-fixing version of the problem can now be defined formally.

PROBLEM: ORDER-FIXING ROBUST $1|r_j|\sum w_j U_j$ (DECISION)
INPUT DATA: $(V, \mathcal{J}, (r, d, w, p, \bar{\delta}, \tau, f), \Xi)$, where $V \in \mathbb{R}$ is a target value, $\mathcal{J}$, a set of jobs characterized by data (r, d, w, p), and for each job j, a maximum additional cost $\bar{\delta}_j$ if j fails, a fixed extra time τ_j needed to fix j, a fixed cost f_j for outsourcing j, and Ξ is an uncertainty set.
QUESTION:

Is there a subset $A \subseteq \mathcal{J}$ of on-time jobs, and a permutation σ of the elements of A such that for any scenario $\xi \in \Xi$, FIXED-ORDER-REPAIRING PROBLEM with data $(V, \mathcal{J}, (r, d, w, p, \bar{\delta}, \tau, f), A, \xi, \sigma)$ has answer yes?

Problem $(\widetilde{P})$ can be formulated in a similar way as what has been done for problem (P). Just like in Section 4.2, let us denote, for any job occurrence $k \in \widetilde{\mathcal{J}}$, by x_k the selection of the kth job occurrence in the non-decreasing order of their deadlines (i.e., 1 if the kth job occurrence is used, 0 otherwise). Variables y_k, z_k for job occurrences and U_j will keep the same meaning as in the previous section.

Again, regarding the set of admissible recourses, the schedule-feasibility property is dealt with by set Y introduced in Section 4.3. Concerning the non-anticipativity property, which stipulates that the recourse action should not contradict a first-stage decision, we impose that $y_k \leq x_k, \forall k \in \widetilde{\mathcal{J}}$. That is, that one may confirm the execution of a job, or fix a job, only if it were actually accepted in the first stage. The set of admissible recourses is then given by the following set: $\widetilde{Y}(\boldsymbol{x}) = \{(\boldsymbol{y}, \boldsymbol{z}) \in Y \mid y_k \leq x_k \quad \forall k \in \widetilde{\mathcal{J}}\}$.

We now detail the different possible decisions. Let $j \in \mathcal{J}$ be a job:

- if $U_j = 1$, then the job is executed tardy and no recourse action may be taken (i.e., $y_k = z_k = 0$)
- if there is k such that $k \in \mathcal{G}_j$ and $x_k = 1$, the job is to be scheduled on time in the first stage:
  - if $y_k = z_k = 1$, job j is executed and fixed on time
  - if $y_k = 1$ and $z_k = 0$, job j is executed on time and a penalty is paid
  - if $y_k = z_k = 0$, job j is outsourced

The objective function can therefore be expressed as follow:

$$\min_{(\boldsymbol{x},\boldsymbol{U}) \in \widetilde{X}} \sum_{j \in \mathcal{J}} [w_j U_j + f_j(1 - U_j)] + \max_{\boldsymbol{\xi} \in \Xi} \min_{(\boldsymbol{y},\boldsymbol{z}) \in \widetilde{Y}(\boldsymbol{x})} R(\boldsymbol{\xi}, \boldsymbol{y}, \boldsymbol{z})$$

where $\widetilde{X}$ denotes the set of feasible first-stage solutions (i.e., the set of solutions which define a sequence of tasks), that is:

$$\widetilde{X} = \left\{ (\boldsymbol{x}, \boldsymbol{U}) \in \{0,1\}^{|\widetilde{\mathcal{J}}|} \times \{0,1\}^{|\mathcal{J}|} \;\middle|\; \sum_{k \in \mathcal{G}_j} x_k + U_j = 1 \quad \forall j \in \mathcal{J} \right\}$$

Again, note that the first-stage decision does not have to be physically feasible in the sense that the optimal solution may be to decide a sequence of jobs in the first stage and to outsource

some of them in the second stage so as to make the schedule feasible.

4.5.2 Relation with problem without order- xation

This section is devoted to show that the first problem yields a lower bound for the second problem, which is an intuitive result: compared to $(\widetilde{P})$, in (P) some decisions are postponed. That means that, for different realizations of the uncertainty those decisions can be different in (P) but must be identical in $(\widetilde{P})$. In that sense, (P) relaxes some of the non-anticipativity constraints of $(\widetilde{P})$.

Remark 12. *Denoting by ( ) the optimal value of problem , the following relation holds:*

$$(P) \quad (\widetilde{P})$$

We now provide an example showing that $(\widetilde{P})$ is a strict relaxation of $(\widetilde{P})$.

Remark 13. *Given one problem instance, optimal solutions to (P) may attain a strictly lower objective value than the optimal solutions to $(\widetilde{P})$.*

Proof. Consider the following instance:

j	r_j	d_j	p_j	τ_j	w_j	$\bar{\delta}_j$	f_j
i	0	6	1	4	100	6	*1*
j	5	8	2	2	100	4	*1*
k	1	9	2	3	100	5	*1*

, $\Gamma = 1$

where the outsourcing of a task is never considered (not affordable) for simplicity. The three jobs can be scheduled on time and it is never optimal to execute a job tardy, even after knowing the penalty (i.e., it is always better to pay the penalty than to execute the job tardy). That being said, it is clear that the optimal sequence is either (i, k, j) or (i, j, k). Since the uncertainty parameter Γ is set to one, exactly one job will be affected by the uncertainty (note that, because we are in a two-stage robust context, the uncertainty can be spread among different random parameters in the worst case, but this does not happen for this instance).

Let us first consider problem $(\widetilde{P})$ where one decides the sequencing of the jobs before knowing the uncertainty. Figure 4.2 depicts the two solutions detailed below. If the decision, in the first stage, implies using sequence i, k, j, then it is only possible to fix k. Thus, the cost of the worst case is given by $\max(\bar{\delta}_i, \bar{\delta}_j, 0) = \max(6, 4, 0) = 6$. If one were to choose sequence i, j, k however, the only fixable task is i which means that the worst-case scenario costs $\max(0, \bar{\delta}_j, \bar{\delta}_k) = \max(0, 4, 5) = 5$. The optimal solution to the overall problem minimizes the worst-case cost, hence $(\widetilde{P}) \ = 5$.

If we consider (P) where one only selects on-time jobs in the first stage however, one can better react to the uncertainty in the second stage. Indeed, if the uncertainty affects job j one is forced to pay $\bar{\delta}_j = 4$ since it is never possible to fix it. However, if the uncertainty affects job i, one can react to that scenario by choosing the sequence i, j, k under which job j can be fixed. If, to the contrary, the uncertainty affects job k, the optimal recourse decision is realised by using the sequence i, k, j under which one can fix job k. This shows easily that, for this problem, the worst case is realised when the uncertainty hits job j. In any way, $(\widetilde{P}) \ > (P) \ \ (5 > 4)$. □ □

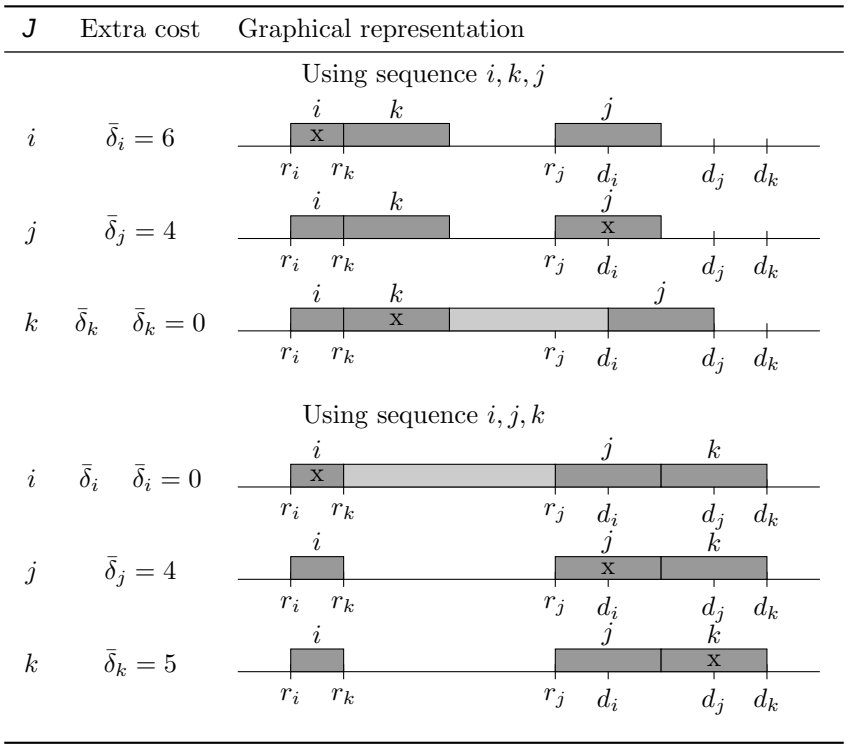


Figure 4.2: Two schedules and the associated extra cost under the failure of each job if the sequence of jobs is fixed before the uncertainty is revealed

From left to right : the failing job, the associated extra cost, a graphical representation of the schedule

4.6 Computational experiments

This section reports the main computational results for the two problems we are addressing. We first give some details about our implementation, and then explain how random instances were generated. We also describe our protocol to compare the exact approach with the finite adaptability methods, which are exact only if the input parameter K is large enough.

4.6.1 Implementation details and experimental setting

All mixed integer linear programs, as well as linear programs inside the column generation procedures, are solved using IBM ILOG Cplex 12.9, through the C callable library, using default parameters and four threads. The generic implementation BapCod Vanderbeck [2005] of the branch-and-price Algorithm 1 is used to optimize models ColGen1 and ColGen2. At each node of the search tree, the linear relaxation of the problem is computed using column generation (Algorithm 2). The pricing sub-problem is solved using the MILP solver. At most one column is added to the master program $[DEP]^R$ at each iteration. To improve the convergence of the column generation procedure, we use stabilization by automatic smoothing of the dual variables of the master program, as described in Pessoa et al. [2018]. When the optimal solution of the corresponding relaxation does not satisfy the integrality requirements of first-stage variables, one

fractional variable is chosen and two child nodes are created in order to exclude its current value from the search space. This variable is chosen to be the closest from 0.5. The open nodes are processed according to the best first rule. The implementation of the branch-and-price algorithm is sequential.

The approach from Subramanyam et al. [2019] (i.e., KAdapt1-b and KAdapt2-b) for solving the finite-adaptability counterpart has been implemented in C++ using the author's code which is publicly available[2].

All our experiments are conducted using a 2 Dodeca-core Haswell Intel Xeon E5-2680 v3 2.5 GHz machine with 128Go RAM running Linux OS, part of the PlaFRIM[3] experimental platform. The resources of this machine are strictly partitioned using Slurm Workload Manager[4] to run several tests in parallel. The resources available for each run (algorithm-instance) are set to 4 threads and a 20 Go RAM limit (we remark that our branch-and-price algorithm does not benefit from parallel processing). This virtually creates six independent machines, each running one single instance at a time.

4.6.2 Instances

The test bed was randomly generated based on the technique used in Dauzère-Pérès and Sevaux [2003] in which the authors generate a random test bed for the deterministic $1|r_j|\sum w_j U_j$ problem. Their approach takes as input three parameters: the number of jobs N, a factor for the dispersion of the release dates R_1 and a factor controlling the dispersion of the deadlines R_2. Having fixed these parameters, we generate, for each of the N jobs, random characteristics defined as follows[5]:

$$p_j \quad \mathcal{U}(1,100) \qquad w_j \quad \mathcal{U}(1,100)$$
$$\bar{\delta}_j \quad \mathcal{U}(1,100) \qquad f_j \quad \mathcal{U}(1,100)$$
$$r_j \quad \mathcal{U}(0,N \quad R_1) \qquad \Delta_j \quad \mathcal{U}(0,N \quad R_2)$$
$$d_j = r_j + p_j + \Delta_j \qquad \tau_j \quad \mathcal{U}(0,\lambda\Delta_j)$$

Here, Δ_j denotes the slack time of jobs j within its time window. Note that the extra time needed to fix one job τ_j is generated depending on an extra parameter λ, fixed, for our experiments, to $\frac{5}{4}$. This implies that it is *generally* feasible to fix a job if it had its whole time window to be scheduled (i.e., if no other job interferes with it). The parameters which were used are combinations of $N \in \{5, 10, 15, 20, 25\}$, $R_1 \in \{5, 10, 20, 30\}$ and $R_2 \in \{5, 10, 20, 30\}$. In total, 480 instances were generated. Yet, each of these instances are parameterized by the uncertainty budget Γ. Throughout our experiments, the value for Γ varied, from 1 to 3 for 5-job instances, from 1 to 7 for 10-job instances, from 1 to 10 for 15, 20 and 25-job instances. Therefore, we compared the two approaches over 3200 instances.

[2] https://github.com/AnirudhSubramanyam/KAdaptabilitySolver

[3] PlaFRIM: Plateforme Federative pour la Recherche en Informatique et Mathematiques (https://www.plafrim.fr/fr/accueil/)

[4] https://slurm.schedmd.com/ (accessed June 2020)

[5] $\mathcal{U}$ denotes the discrete uniform distribution law

4.6.3 Protocol for comparing the two solution methods

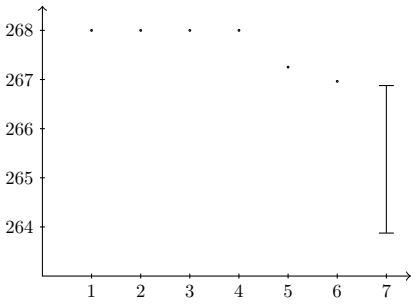
The finite adaptability method solves the problem exactly only if parameter K is large enough; otherwise it solves an approximation which is tighter and tighter as its parameter K grows. For this reason, we must be careful when comparing its numerical performance with the one of the convexification approach. For a given problem (P), let (P_K) be its approximation as a K-adaptability problem. We denote by $(\ )$ the optimal solution of problem (or the best known upper bound in case of reached time limit) and by $t(\ )$ the computation time to solve problem . An attractive case to compare the finite adaptability and the exact approach is when $(P) = (P_K)$, since it amounts to comparing two exact methods. However, large values of K lead to intractable models, so we need to find the smallest value for K which fulfills this condition. More formally, we are interested in the following problem : $K = \min\{K : (P_K) = (P), K \in \mathbb{N}\}$. This problem is feasible and has an upper bound equal to $\dim \Xi + 1 = |J| + 1$ (Hanasusanto et al. [2015]). Note that the a priori knowledge of the optimal value (P) is assumed. If this assumption is not satisfied, we do not have a practical method to find K, since a local minimum of function $K \mapsto (P_K)$ sometimes fails to be global. This is illustrated in Figure 4.3, which reports the optimal objective value for a specific 15-job instance of $(\widetilde{P})$ for various values of K. We can see that from the 1-adaptability to the 4-adaptability, the optimal value does not change while it does for greater values of K. This shows that guessing the value K, for which the approximation is, in fact, an exact solution, is hard and to our knowledge there is no straightforward stopping criterion for the search for K. As a matter of fact, for that specific instance, we are unable to conclude if $K = 7$ or if $K > 7$ since we were unable to solve it within one hour. That particular observation holds for the two problems (P) and $(\widetilde{P})$.

We run our algorithms under a given time limit and not all instances are solved to optimality, thus (P), (P_K) and their fixed-order counterparts are sometimes approximated. To overcome this difficulty, we focus on a slightly different problem: find the smallest value for K which, under a given time limit T, yields an objective function at least as good as the solution of the exact approach. Formally, we estimate K by

$$\widehat{K} = \min \left\{ K : \begin{array}{ll} (P_K) & (P), \\ t(P) & T, \\ t(P_K) & T, \\ K \in \mathbb{N} & \end{array} \right\}$$

In our experiments, the search for $\widehat{K}$ is done iteratively starting from $K = 1$ and increasing K by one unit until one of the four conditions is reached:

- (P_K) (P), $t(P)$ T and $t(P_K)$ T : the two problems were solved optimally and we set $\widehat{K} = K$ (in this case, the equality of the objectives hold and the approximation is tight: $\widehat{K} = K$);
- (P_K) (P), $t(P) > T$ and $t(P_K)$ T : the exact approach could not achieve and/or prove optimality, and (P) equals the best upper bound found. We set $\widehat{K} = K$ and we know that $\widehat{K}$ K ;
- $t(P)$ T and $t(P_K) > T$: the finite adaptability approach could not achieve and/or



K	Objective	Gap
1	269.000	
2	269.000	
3	269.000	
4	269.000	
5	268.252	
6	267.963	
7	267.888	1.129

Figure 4.3: A K-adaptability plateau for a 15-jobs instances.
The optimal objective value does not change between the 1-adaptability and the 4-adaptability, we were unable to solve the instance using the 7-adaptability within one hour (the vertical bar represents the optimality gap).

prove optimality, the search for $\widehat{K}$ is stopped since increasing K typically increases the computation time to solve (P_K). We set $\widehat{K} = K$ and we know that $\widehat{K} \quad K$;

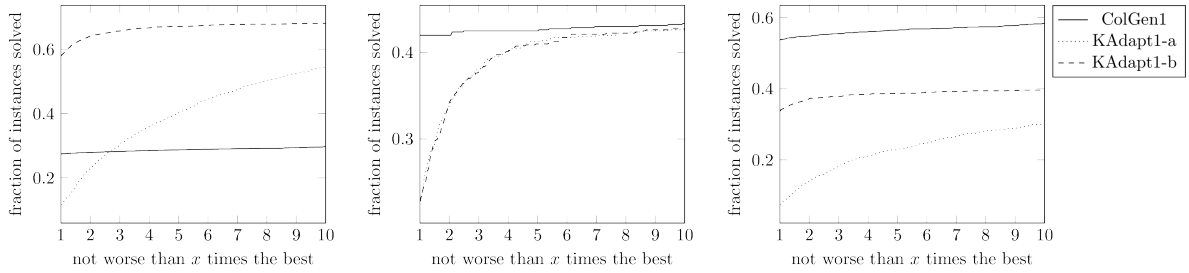
– $t(\mathsf{P}) > T$ and $t(\mathsf{P}_K) > T$: none of the two problems could be solved to proven optimality, the search for $\widehat{K}$ is stopped and the two methods are considered to perform as badly as the other. We set $\widehat{K} = K$ and we know that $\widehat{K} \quad K$.

Note that, as K is typically unknown, comparing (P) with (P_K) or $(\mathsf{P}_{\widehat{K}})$ in fact gives an advantage to the finite adaptability.

4.6.4 Comparison of the approaches for problem without order- xation

In Table 4.1, we present a comparison between finite adaptability from Section 4.4.1 (columns KAdapt1-a and KAdapt1-b) and the exact approach from Section 4.4.2 (columns ColGen1). We use the experimental protocol described above with a time limit $T = 1$ hour for each run. The two first columns describe the main characteristics of the instances considered (number of jobs and value of Γ). The left-hand part of the table gathers the percentage of instances that were not solved to optimality within the time limit. Then, the average computing time is reported (instances for which the time limit is reached count for 3600 seconds). In the right-hand part of the table, we finally reported the percentage of times in which one solution was found to be the fastest among the three. The same data are illustrated as performance profiles in Figure 4.4.

The finite adaptability approach KAdapt1-a solves all five-job instances, but fails to solve 11.25% of the instances for 10 jobs and $\Gamma = 2$. Less than 15% of the 25-job instances instances are solved by this method when $\Gamma \quad 4$. Indeed, we have noticed that the hardest instances correspond to those having a value of Γ such that the ratio $\Gamma/|J|$ is around 0.3-0.4. For very small values of Γ the problem often becomes easy since a small number of critical jobs have to be found. For large values of Γ, to the contrary, the problem almost reduces to the deterministic problem where all the jobs are penalized. The in-between instances are the most challenging. Regarding KAdapt1-b, our results show that the method proposed in Subramanyam et al. [2019] is more efficient than the MILP formulation. Indeed, the average CPU time for KAdapt1-b is typically smaller than the time spent solving the MILP model for KAdapt1-a. Moreover, it solves significantly more hard instances than the MILP approach (see Figure 4.4c). The branch-and-price algorithm ColGen1 is



(a) All instances (b) 25-job instances (c) Instances with $|J|/4$

Figure 4.4: Performance profiles Dolan and Moré [2002] for different sets of instances. Each curve is associated with one method, and shows the fraction of instances it solves not slower than the value of the abscissa times the time required for the fastest approach

able to solve all instances up to 20 jobs to optimality, and more than 88% of the 25-job instances. When finite adaptability is able to find the optimal solution, it is generally faster than ColGen1. Note that the reported computing times are those of the *last* run of KAdapt1 during the search for $\hat{K}$ (those can be obtained only if one is able to "guess" value of $\hat{K}$ beforehand, which is not the case in a practical context).

Table 4.2 shows the percentage of 25-job instances for which each method could find a feasible solution within the time limit T although it was not able to prove its optimality. KAdapt1-a always finds a feasible solution while it is clearly not the case for the convexification approach. Once again, note that the results for KAdapt1-a are obtained with the first value of K for which the execution time exceeded T, which may not be equal to K . This means that the cost reported is an upper bound of the actual cost of the first-stage solution found by K-adaptability (since the recourse used is heuristic if K is not large enough). For these instances, KAdapt1-a always finds a feasible solution but with a very large optimality gap: the gap between the lower and upper bounds of the MILP solver at the time limit is larger than 70% on average. This is partly explained by the poor linear relaxation of the MILP model. Conversely, the branch-and-price algorithm ColGen1 based on the convexification approach does not always find a feasible solution, but when it does, the optimality gap is often small (4.5% on average). Similarly, KAdapt1-b always finds a feasible solution within the given time limit, except for a pathological case of 25-jobs and $\Gamma = 1$.

In Table 4.3, we study the values of K that are needed to obtain the optimal solution with the K adaptability method. We considered only the approach KAdapt1-a since this method has been shown to be more efficient for our case. For this purpose, we report for every value of K from 1 to K , the average gap between the value found by KAdapt1-a and the exact method ColGen1. We also compare computation times of KAdapt1-a and ColGen1, according to the different values of K. An important information gathered from the table is that a very large proportion of instances can be solved to optimality within one hour with a value of $K = 1$. This means that for many instances, the so-called static model where the recourse actions are decided a priori is sufficient to solve the problem.

It can also be noted that K adaptability with a small value of K can be a good heuristic: for the 16 instances where $\hat{K}$ $= 5$, setting K to 1 produces a gap of less than 7%, for computation times often two order of magnitude smaller than the time required to solve the problem optimally using the branch-and-price algorithm. This table also shows the high sensitivity of the

K adaptability approach with respect to parameter K. For example, let us consider instances with $\hat{K} = 2$. When $K = 1$, 587 instances can be solved within one hour, whereas incrementing the value of K to 2 allows solving only 110 instances within the same time limit. KAdapt1-a appears, at first glance, to perform surprisingly better when $\hat{K}$ increases. Indeed, when $\hat{K} = 5$ and $K = 3$, and when $\hat{K} = 6$ and $K = 4$, its computation time looks significantly smaller than the one for ColGen1. But this anomaly is explained by the fact that only the instances that could be solved using KAdapt1-a with $K = 5$ and $K = 6$, respectively, are reported in these sections of the table. They are very likely to be well-suited for this approach, which explains the very good results when the value of K is smaller. Notice that we could determine the value of K for 2231 out of the 3200 instances, leaving this question open for the 969 others.

Finally, looking at the last column of table 4.3, one can see that KAdapt1-a, when $K = 1$, is significantly faster than ColGen1. In this case, the MILP model simplifies to a static robust optimization model (thanks to the structure of constraint (4.30)), which explains the very good performance. However, for the more complex settings, ColGen1 outperforms KAdapt1 by one to two orders of magnitude.

4.6.5 Comparison of the approaches for the order- xing problem

Table 4.4 compares computation times for approaches KAdapt2-a, KAdapt2-b and ColGen2. The values reported for KAdapt2-a and KAdapt-b are obtained with parameter $K = \hat{K}$ for each instance.

We can see that problem $(\widetilde{P})$ is much more challenging to solve than (P), since the K adaptability methods reach limitations for 10-job instances while the branch-and-price algorithm ColGen2 is unable to solve some 15-job instances. This is mainly explained by the relatively large number of variables in the models. Indeed, while the number of first-stage variables was $O(|J|)$ for problem (P), it is now $O(|\widetilde{J}|) = O(|J|^2)$. Hence, there is a significant number of additional variables subject to integrality constraints in problem $(\widetilde{P})$ compared to (P).

Table 4.5 shows the computation time ratio between KAdapt2-a and ColGen2. As for problem (P), we can see that the closer K gets to $\hat{K}$, the faster the branch-and-price algorithm becomes compared to K-adaptability.

We also studied the increase of the objective function when one has to decide the sequence of the selected jobs before uncertainty is revealed. In Table 4.6, for several sizes of instances and several values of Γ, we report the average costs related to problem (P) (column Free) and $(\widetilde{P})$ (column fixed order), respectively.

Our experiments show that fixing the sequence of jobs beforehand only leads to a marginal increase of the cost on average. The largest gap we obtained was 0.26% for instances with ten jobs.

As a conclusion, it appears that for this problem, the cost to pay to keep the order fixed in the first-stage is not the cost of the solutions itself, but the practical difficulty to solve the optimization problem with state-of-the-art algorithms.

4.7 Conclusion

In this chapter, we have described a robust version of the classical one-machine scheduling problem where one minimizes the weighted number of tardy jobs. Although solving general robust integer programs with integer recourse is typically Σ_P^2-hard, we were able to show that this problem is NP-complete, and proposed two solution approaches: an exact reformulation which can be solved by means of the branch-and-price algorithmic procedure, and a (MILP) conservative approximation. Our computational experiments show that this problem is hard to solve in practice, since state-of-the-art methods may fail to solve 25-job instances in one hour.

Regarding the exact method we proposed, we think that the development of good heuristic procedures for the pricing problem may substantially improve the computing times. As for the conservative approximation, its main drawback is its poor linear relaxation, which is a known issue in the literature. We also have investigated another version of the problem where the sequencing decisions from the first stage cannot be modified. It appears that this version of problem is harder than the first: some 15-job instances are left unsolved by both approaches.

Acknowledgments

Experiments presented in this chapter were carried out using the PLAFRIM experimental testbed, being developed under the Inria PlaFRIM development action with support from Bordeaux INP, LABRI and IMB and other entities: Conseil Régional d'Aquitaine, Université de Bordeaux, CNRS and ANR in accordance to the programme d'investissements d'Avenir (see `https://www.plafrim.fr/`).

Table 4.1: CPU execution times for solving problem (P)

		Unsolved within $T = 1$ hour (%)			Average CPU time (s.)			Fastest approach (%)		
$\|J\|$	Γ	KAdapt1-a	Kadapt1-b	ColGen1	KAdapt1-a	KAdapt1-b	ColGen1	KAdapt1-a	KAdapt1-b	ColGen1
5	1	0	0	0	0	0	2	10	90	0
	2	0	1	0	0	0	2	4	96	0
	3	0	0	0	0	0	2	1	99	0
Avg. $\|J\| = 5$		0	0	0	0	0	2	5	95	0
10	1	6	12	0	85	59	13	11	76	12
	2	11	24	0	49	30	15	18	64	19
	3	5	10	0	17	1	11	5	85	10
	4	1	2	0	0	1	8	1	96	2
	5	1	2	0	7	0	7	1	96	2
	6	0	0	0	0	0	7	1	99	0
	7	0	0	0	0	0	7	1	99	0
Avg. $\|J\| = 10$		4	7	0	22	13	10	6	88	7
15	1	35	28	0	443	207	43	8	50	42
	2	57	69	0	452	27	69	4	28	69
	3	46	49	0	16	1	64	1	50	49
	4	29	29	0	55	0	47	4	68	29
	5	12	12	0	0	0	29	6	81	12
	6	10	10	0	0	0	23	2	88	10
	7	2	2	0	0	0	21	4	94	2
	8	0	0	0	0	0	17	4	96	0
	9	0	0	0	0	0	16	1	99	0
	10	0	0	0	0	0	16	1	99	0
Avg. $\|J\| = 15$		19	20	0	97	24	35	3	75	21
20	1	66	45	0	693	184	118	2	44	54
	2	88	86	0	171	39	190	4	12	84
	3	86	91	0	306	42	273	5	6	89
	4	71	76	0	132	20	332	6	19	75
	5	55	56	0	20	1	346	5	39	56
	6	35	35	0	0	0	269	16	49	35
	7	20	20	0	0	0	188	20	60	20
	8	5	5	0	0	0	127	31	64	5
	9	0	0	0	0	0	67	34	66	0
	10	0	0	0	0	0	46	28	72	0
Avg. $\|J\| = 20$		43	42	0	132	29	196	15	43	42
25	1	82	59	9	384	345	375	8	30	62
	2	95	95	12	78	45	623	11	9	79
	3	91	92	16	5	0	629	15	16	69
	4	78	78	19	0	0	613	18	25	56
	5	66	66	21	0	1	538	25	29	46
	6	57	57	20	0	1	534	29	29	41
	7	39	39	18	0	0	442	33	38	30
	8	31	31	12	0	1	442	38	37	26
	9	15	15	10	0	1	426	50	37	13
	10	6	6	5	0	2	286	59	35	6
Avg. $\|J\| = 25$		56	54	14	47	40	491	29	29	43

From left to right : the number of jobs, the uncertainty budget, the average computation time (when less than $T = 1$ hour) for the K-adaptability approach, the convexification-based branch-and-price algorithm, the percentage of times one method was found to be the most efficient and the percentage of instances which could not be solved within $T = 1$ hour. For the sake of readability, all numbers are rounded to the closest integer.

Table 4.2: Feasible solutions found for (P), over instances that could not be solved to optimality by the method within the time limit $T = 1$ hour

# Jobs	Γ	Feasible solutions found (%) KAdapt1-a	KAdapt1-b	ColGen1
25	1	100.00	100.00	28.57
	2	100.00	100.00	30.00
	3	100.00	100.00	15.38
	4	100.00	100.00	6.67
	5	100.00	100.00	5.88
	6	100.00	100.00	6.25
	7	100.00	100.00	0.00
	8	100.00	100.00	10.00
	9	100.00	100.00	25.00
	10	100.00	100.00	25.00

From left to right : the number of jobs, the uncertainty budget and the percentage of instances for which a feasible solution could be found within $T = 1$ hour over the instances which could not be solved optimally within $T = 1$ hour. Numbers are rounded to the closest integer.

Table 4.3: The cost of approximating with finite adaptability for problem (P)

$\hat{K}$	K	# Instances	Approximation gap (%)	Time ratio
1	1	1989	0	0.03
2	1	587	6.6	0.01
	2	110	0	5.23
3	1	368	0.00	0.03
	2	368	1.43	6.00
	3	90	0	14.59
4	1	123	6.3	0.01
	2	123	2.06	0.09
	3	123	0.6	14.23
	4	32	0	13.27
5	1	16	6.85	0.01
	2	16	2.25	0.04
	3	16	0.67	0.82
	4	16	0.24	34.76
	5	3	0	29.28
6	1	3	1.92	0.01
	2	3	1.7	0.02
	3	3	0.58	0.05
	4	3	0.13	0.31
	5	3	0.02	5.87
	6	2	0	2.06

From left to right : the value of $\hat{K}$ (*i.e.* the value of K required for K-adaptability to be equivalent to (P) or a lower bound on this value), the value of K which was used, the number of instances with this value of $\hat{K}$ which were solved using Kadapt1 with that value of K within $T = 1$ hour, the approximation gap computed as $jKAdapt1 \quad ColGen1\ j/jColGen1\ j$, the computation time ratio computed as $t(KAdapt1)/t(ColGen1)$.

Table 4.4: Computation times for solving problem ($\widetilde{P}$)

		Unsolved within 1 hour (%)			Average CPU time (s.)			Fastest approach (%)		
$\|J\|$	Γ	KAdapt2-a	KAdapt2-b	ColGen2	KAdapt2-a	KAdapt2-b	ColGen2	KAdapt2-a	KAdapt2-b	ColGen2
5	1	0	0	0	0	0	1	12	87	1
	2	0	0	0	0	0	1	14	86	0
	3	0	0	0	0	0	1	7	93	0
Avg. $\|J\| = 5$		0	0	0	0	0	1	11	88	0
	1	0	11	0	14	27	28	22	72	5
	2	1	22	0	22	18	24	28	59	14
	3	1	9	0	9	4	11	16	78	6
	4	0	2	0	43	2	8	6	91	2
	5	0	1	0	37	40	4	12	86	1
	6	0	0	0	0	0	3	6	94	0
	7	0	0	0	0	0	3	8	92	0
Avg. $\|J\| = 10$		0	7	0	18	13	12	14	82	4
15	1	11	18	8	191	118	243	14	61	25
	2	36	55	16	212	280	219	25	27	48
	3	25	38	12	154	0	158	19	46	35
	4	19	26	8	131	0	169	21	56	23
	5	9	10	4	24	0	77	18	74	9
	6	5	6	4	8	0	47	20	76	5
	7	2	2	2	0	0	61	23	74	2
	8	0	0	0	0	0	50	19	81	0
	9	0	0	0	0	0	15	12	88	0
Avg. $\|J\| = 15$		12	17	6	80	44	115	19	65	16

From left to right: the number of jobs, the uncertainty budget, the average computation time (when less than $T = 1$ hour) for each method, the percentage of times one method was found to be the most efficient and the percentage of instances which could not be solved within the time limit $T = 1$ hour. All numbers are rounded to the closest integer.

Table 4.5: The cost of approximating with finite adaptability for problem $(\widetilde{P})$

$\hat{K}$	K	# Instances	Approximation gap (%)	Time ratio
1	1	1165	0	0.23
2	1	87	4.88	0.06
	2	87	0	0.11
3	1	79	6.37	0.02
	2	76	0.97	0.09
	3	65	0	3.74
4	1	93	7.15	0.04
	2	93	1.65	0.09
	3	73	0.4	2.01
	4	59	0	5.79
5	1	40	7.93	0
	2	40	2.72	0.03
	3	40	0.83	0.53
	4	40	0.19	8.75
	5	14	0	14.17
6	1	14	5.84	0.01
	2	14	2.98	0.02
	3	14	1.12	0.09
	4	14	0.41	0.81
	5	14	0.09	13.24
	6	6	0	35.19
7	1	2	0.37	0.01
	2	2	0.37	0.02
	3	2	0.37	0.14
	4	2	0.37	0.65
	5	2	0.14	5.52
	6	2	0.03	78.13

From left to right : the value of $\hat{K}$ (*i.e.* the value of K required for K-adaptability to be equivalent to (P) or a lower bound on this value), the value of K which was used, the number of instances with this value of $\hat{K}$ which were solved using Kadapt2 with that value of K within $T = 1$ hour, the approximation gap computed as $jKAdapt2 \quad ColGen2\ j/jColGen2\ j$, the computation time ratio computed as $t(KAdapt2)/t(ColGen2)$.

# Jobs	Γ	Objective cost Free (ColGen1)	Fixed order (ColGen2)	Gap (%)	N. Instance
5	1	70.39	70.40	0.00	80
5	2	75.27	75.28	0.01	80
5	3	75.94	75.94	0.00	80
10	1	144.76	145.14	0.26	80
10	2	165.92	166.21	0.18	80
10	3	171.73	171.80	0.04	80
10	4	173.24	173.26	0.01	80
10	5	173.61	173.61	0.00	80
10	6	173.70	173.70	0.00	80
10	7	173.70	173.70	0.00	80
15	1	192.83	193.32	0.25	74
15	2	232.46	233.06	0.26	67
15	3	248.97	249.57	0.24	70
15	4	253.39	253.78	0.15	74
15	5	254.77	254.91	0.05	77
15	6	255.35	255.37	0.01	77
15	7	255.74	255.74	0.00	78
15	8	256.98	256.98	0.00	80
15	9	256.98	256.98	0.00	80
15	10	256.98	256.98	0.00	80

Table 4.6: Fixed-order solutions cost analysis

From left to right: the number of jobs, the uncertainty budget, the average objective costs of free solutions and fixed-ordered solutions, the relative gap between the two, the number of instances which where accounted for in the computation (i.e., instances which could be solved within the time limit $T = 1$ hour for both problems).

CHAPTER 5

Mixed-integer problems with binary uncertainty

Henri Lefebvre, Enrico Malaguti, Michele Monaci[1]

In this chapter, we study Adjustable Robust Optimization (ARO) problems with discrete uncertainty. Under a very general modeling framework, we show that such two-stage robust problems can be exactly reformulated as ARO problems with objective uncertainty only. This reformulation is valid with and without the fixed recourse assumption and is not limited to continuous wait-and-see decision variables, unlike most of the existing literature. We then discuss how to apply the reformulation on a variant of the Multiple Knapsack Problem where each item's weight is regarded as an unknown parameter. The latter problem is also shown to include Σ_2^P-hard problems.

Additionally, we introduce an enumerative algorithm akin to a Benders branch-and-cut scheme for which we study the asymptotic convergence. Finally, we report extensive computational results on instances of the two considered problems, demonstrating the effectiveness of the approach.

5.1 Introduction

In this chapter, we consider the following underlying optimization problem, where it is assumed that $\boldsymbol{H}$ is a matrix which cannot be known at decision time.

$$\min \boldsymbol{c}^T\boldsymbol{x} + \boldsymbol{d}^T\boldsymbol{y} \tag{5.1}$$

$$\text{s.t.} \boldsymbol{T}\boldsymbol{x} + \boldsymbol{H}\boldsymbol{y} \quad \boldsymbol{f} \tag{5.2}$$

$$\boldsymbol{x} \; 2 \; X \tag{5.3}$$

$$\boldsymbol{y} \; 2 \; Y \tag{5.4}$$

$$\boldsymbol{0} \quad \boldsymbol{y} \quad \boldsymbol{u} \tag{5.5}$$

[1] The content of this chapter has been submitted to *INFORMS Journal on Computing* and is currently under revision.

Here, $\boldsymbol{x}$ denotes the vector of here-and-now decisions, $\boldsymbol{y}$ represents wait-and-see decisions that can be taken after uncertainty reveals, and $\boldsymbol{c}, \boldsymbol{d}, \boldsymbol{T}, \boldsymbol{H}, \boldsymbol{f}$ and $\boldsymbol{u}$ are real vectors and matrices of appropriate dimension. The objective function includes a cost for both here-and-now and wait-and-see decisions; the corresponding variables are coupled by means of linking constraints (5.2). Constraints (5.3) and (5.4) impose that $\boldsymbol{x}$ and $\boldsymbol{y}$ belong to some polyhedral set; these constraints may also include integrality requirements on some variables, if any. Moreover, we assume that matrix $\boldsymbol{H}$ is known to take a finite set of values, noted $\mathcal{H}$. Let us denote by $Y(\hat{\boldsymbol{x}}, \hat{\boldsymbol{H}}) = \{\boldsymbol{y} \in Y : \hat{\boldsymbol{H}}\boldsymbol{y} \geq \boldsymbol{f} - \boldsymbol{T}\hat{\boldsymbol{x}}, \boldsymbol{0} \leq \boldsymbol{y} \leq \boldsymbol{u}\}$ the set of feasible wait-and-see decisions for a given here-and-now decision $\hat{\boldsymbol{x}} \in X$ and random outcome $\hat{\boldsymbol{H}} \in \mathcal{H}$ of $\boldsymbol{H}$.

Already in the case of min-max optimization and objective discrete uncertainty, it has been shown Buchheim and Kurtz [2018] that the problem is NP-hard as soon as the here-and-now decisions are binary and the uncertainty set contains exactly two scenarios. Since min-max-min problems are a generalization of min-max optimization, this complexity result trivially extends and several approaches have been designed to approximately or exactly address the general min-max-min case.

Assuming that the convex hull of the uncertainty set is known, one considerably simplifies the problem by replacing Ξ with $\text{conv}(\Xi)$, leading to a relaxation of the original problem. In such cases, typical approaches designed for continuous uncertainty sets as described in Chapter 2 may be employed.

To the best of our knowledge, however, no viable solution approach has been introduced in the literature for the case where the wait-and-see decisions are mixed-integer and uncertainty is discrete and affects the problem constraints. This work is aimed at bridging this gap between objective-uncertain problems and constraint-uncertain problems with mixed-integer wait-and-see decisions, and provides the following main contributions:

- In Section 5.2, we introduce our modeling framework and discuss its generality. We show that virtually any linearly-constrained Adjustable Robust Problem with discrete uncertainty can fit our framework. Among others, we emphasize that our model includes problems with and without the fixed recourse assumption. Moreover, we allow for both mixed-integer first- and second-stage decisions.

- In Section 5.3, we introduce an ARO problem which is a natural extension of the well-known Multiple Knapsack Problem (MKP) where each item's weight is uncertain. We show that its ARO counterpart includes Σ_2^P-hard problems.

- In Section 5.4, we present our main theoretical results. We prove the general validity of a non-trivial reformulation of ARO problems with discrete uncertainty as objective-uncertain ARO problems. This reformulation is based on polyhedral results and Lagrangian duality.

- In Section 5.5, we extend a recent contribution (Kämmerling and Kurtz [2020]) from the ARO literature to the mixed-integer case while it was originally designed for objective-uncertain problems with binary first-stage decisions only. We also explicitly treat the case of ARO problems which are infeasible, or which do not have complete recourse (i.e., for which $\forall \boldsymbol{x} \in X, \exists \boldsymbol{\xi} \in \Xi, Y(\boldsymbol{x}, \boldsymbol{\xi}) = \emptyset$).

– In Section 5.6, we report computational results for this class of hard optimization problems, showing that our approach is able to solve medium size instances within a reasonable computing time.

5.2 Problem modeling

5.2.1 Uncertainty model

Our modelling of uncertainty is based on the following assumption.

Assumption M. *For all $i = 1, ..., m_Y$ and all $j = 1, ..., n_Y$, let $\underline{h}_{ij}$ and $\bar{h}_{ij}$ be two real numbers such that $\underline{h}_{ij} \quad \bar{h}_{ij}$. The set H of possible outcomes for $\boldsymbol{H}$ is such that $H \quad f\hat{\boldsymbol{H}} : (\hat{h}_{ij} = \underline{h}_{ij} \_ \hat{h}_{ij} = \bar{h}_{ij}), i = 1, ..., m_Y, j = 1, ..., n_Y g$.*

To ease the presentation, we introduce a binary set $\Xi \quad f0, 1g^{m_Y \quad n_Y}$ used to encode the combinatorial aspects of H as follows: for all $\boldsymbol{\xi} \; 2 \; \Xi$ and any (i, j), $\xi_{ij} = 0$ if $\hat{h}_{ij} = \underline{h}_{ij}$ and $\xi_{ij} = 1$ if $\hat{h}_{ij} = \bar{h}_{ij}$. For a given $\hat{\boldsymbol{x}} \; 2 \; X$ and $\hat{\boldsymbol{H}} \; 2 \; H$, with a small abuse of notation we denote set $Y(\hat{\boldsymbol{x}}, \hat{\boldsymbol{H}})$ as $Y(\hat{\boldsymbol{x}}, \hat{\boldsymbol{\xi}})$ for an appropriate $\hat{\boldsymbol{\xi}} \; 2 \; \Xi$. Using this notation, set $Y(\hat{\boldsymbol{x}}, \hat{\boldsymbol{\xi}})$ includes all those elements fulfilling constraints (5.4), (5.5), and

$$\sum_{j=1}^{n_X} t_{ij} \, \hat{x}_j + \sum_{j=1}^{n_Y} \left(\underline{h}_{ij} y_j + (\bar{h}_{ij} \quad \underline{h}_{ij}) \, \hat{\xi}_{ij} \, y_j \right) \quad f_i \qquad i = 1, ..., m_Y \tag{5.6}$$

We assume that wait-and-see decisions are bounded, as it happens in practical applications. In addition, as typically done in adjustable robust optimization, $Y(\boldsymbol{x}, \boldsymbol{\xi})$ is supposed to be non-empty for all $\boldsymbol{x} \; 2 \; X$ and all $\boldsymbol{\xi} \; 2 \; \Xi$. As a consequence, we may restrict our attention to the case in which X is bounded, since otherwise the problem would reduce to select here-and-now decisions producing an infinitely negative cost.

Accordingly, the class of problems which we are addressing can be reformulated as follows:

$$\min_{\boldsymbol{x} 2 X} \left\{ \boldsymbol{c}^T \boldsymbol{x} + \max_{\boldsymbol{\xi} 2 \Xi} \min_{\boldsymbol{y} 2 Y(\boldsymbol{x}, \boldsymbol{\xi})} \boldsymbol{d}^T \boldsymbol{y} \right\} \tag{5.7}$$

In the robust optimization terminology, matrix $\boldsymbol{H}$ is often called the *recourse* matrix. The class of problems we consider in this paper can therefore be summarized as *adjustable robust problems with mixed-integer rst and second stage, uncertain recourse matrix and binary uncertainty*.

5.2.2 Expressiveness of our model

In this section, we show that Assumption M can be done without loss of generality. In addition, we discuss how a large variety of uncertain problems can be cast in our framework without introducing further assumptions.

More than two possible outcomes Assume that a generic coefficient, say h_{ij}, has more than two possible outcomes. Let $R > 2$ be this number and denote by $\hat{h}^1_{ij}, ..., \hat{h}^R_{ij}$ the possible values, sorted by increasing order. In the i-th constraint, we replace variable y_j by R additional variables $y^1_{ij}, ..., y^R_{ij}$, the r-th associated with an uncertain coefficient h^r_{ij} having $\underline{h}^r_{ij} = 0$ and $\bar{h}^r_{ij} = \hat{h}^r_{ij}$.

In order to impose that exactly one value is selected by the uncertainty for coefficient h_{ij}, the following constraint

$$\sum_{r=1}^{R} \xi_{ij}^r = 1 \tag{5.8}$$

has to be added to the definition of Ξ. Finally, the relationship between the original y_j and the additional y_{ij}^r variables can be enforced by imposing the following constraints

$$y_j = y_{ij}^r \quad r = 1, ..., R \tag{5.9}$$

Notice that constraints (5.9) can simply be added to the definition of set Y.

“ ”-inequalities and equalities In our definition of the problem, all linking constraints (5.2) are assumed to be written in the form. Since we make no assumption on the sign of coefficients of $\hat{H}$, any inequality can be rewritten in form.
As to equations, we assume they are replaced by a pair of inequalities before the reformulation is applied. Let i_1 and i_2 the indices of the inequalities derived from a given equation and notice that each variable y_j has, in these constraints, the same coefficients but with complementary signs: $\bar{h}_{i_1 j} = \underline{h}_{i_2 j}$ and $\underline{h}_{i_1 j} = \bar{h}_{i_2 j}$. Hence, consistency in the realization of the same parameter can be enforced by the following constraint

$$\xi_{i_1 j} + \xi_{i_2 j} = 1 \tag{5.10}$$

More in general, our framework allows modelling of situations in which the realization of a pair of coefficients of $\hat{H}$ is correlated; for example, if the first coefficient takes its smallest value, then the second assumes its smallest value as well, and vice-versa. These situations can be handled by adding suitable constraints to the definition of Ξ.

Right-hand-side uncertainty Though our uncertainty model assumes recourse matrix uncertainty, it is clear that right-hand-side uncertainty is comprised within our framework. In other words, it can be used for modelling contexts with and without the fixed recourse assumption. To see this, consider one constraint whose right-hand-side may not be known, i.e., assume that f_i is replaced by $\underline{f}_i + (\bar{f}_i \quad \underline{f}_i)\xi_i^0$ in (5.6) with $\underline{f}_i \quad \bar{f}_i$ for some (unknown) binary ξ_i^0. Then, clearly, adding one decision variable y_{n_Y+1} fixed to 1 which multiplies ξ_i^0 turns our problem in the desired form.

Objective uncertainty Finally, it is well known (see, e.g., Bertsimas and Sim [2004]) that assuming full knowledge of the objective function is without loss of generality. Indeed, uncertainty in vector $\boldsymbol{d}$ in (5.1) can be easily handled by introducing an additional (uncertain) constraint that defines the objective function value. Overall, this shows that our modelling approach defines a completely general setting.

5.3 Example: The Multiple Knapsack Problem

In this section, we give an example of application that fits our framework and analyze its theoretical complexity. We consider a variant of the Multiple Knapsack Problem (MKP). In the deterministic MKP, given a collection of items with associated weights and profits, one should decide a subset of items to be packed inside bins of fixed capacities. We consider here a natural adaptation of the MKP where here-and-now decisions select some items, and dispatch each item to one knapsack. At a second stage, the exact weight of each item is revealed and, for each knapsack, a subset of items having maximum profit is determined while respecting the capacity constraint. We note that, though the here-and-now decisions do not have explicit costs in the objective, not all here-and-now decisions are equivalent for the second stage. For example, while assigning all items to one knapsack is a feasible here-and-now decision, this policy may be sub-optimal with respect to the wait-and-see problem which, therefore, would optimize over a single knapsack only.

Our first theorem states that the adjustable robust variant of the MKP is at least Σ_2^P-hard. To show this result, we formally give the definition, as decision problems, of the Knapsack Problem (KP), the Knapsack Interdiction Problem (KIP) and the Adjustable Robust Multiple Knapsack Problem (ARMKP). We show in Theorem 3 that any instance of KIP, which has been shown to be Σ_2^P-hard (see Caprara et al. [2013]), can be transformed in polynomial time into an instance of the ARMKP.

PROBLEM: KNAPSACK PROBLEM (DECISION)
INPUT DATA:(V, I, w, p, W), where V 2 R is a target value, I is a finite countable set, $w : I$! R_+ and $p : I$! R_+ are two functions and W 2 R_+.
QUESTION: Is there a set J I such that $\sum_{i2J} w(i)$ W and V $\sum_{i2J} p(i)$?

PROBLEM: KNAPSACK INTERDICTION PROBLEM (DECISION)
INPUT DATA: $(V, I, v, w, p, \Gamma, W)$, where V 2 R is a target value, I is a finite countable set, $v : I$! R_+, $w : I$! R_+ and $p : I$! R_+ are given functions, Γ 2 R_+ and W 2 R_+.
QUESTION: Is there a set S I such that $\sum_{i2S} v(i)$ Γ and KNAPSACK PROBLEM with data (V, InS, w, p, W) has an answer no?

PROBLEM: ADJUSTABLE ROBUST MULTIPLE KNAPSACK PROBLEM (DECISION)
INPUT DATA: $(V, I, K, \underline{w}, \overline{w}, p, \Xi, W)$, where V 2 R is a target value, I and K are two finite countable sets, $\underline{w} : I$! R_+, $\overline{w} : I$! R_+ and $p : I$! R_+ are given functions ($\underline{w}$ $\overline{w}$), Ξ $f0, 1g^{jIj}$ and $W : K$! R_+ is a given function.
QUESTION: Is there a partition of I into jKj subsets $Q_1, ..., Q_{jKj}$, such that 8ξ 2 Ξ (i) there exists $v_1, ..., v_k$ 2 R_+ $(k = 1, ..., jKj)$ with $\sum_{k=1}^{jKj} v_k = V$ and, (ii) for $k = 1, ..., jKj$, KNAPSACK PROBLEM with data $(v_k, Q_k, w, p, W(k))$ has an answer yes, where $w : I$! R_+ is a function defined as $w(i) = \underline{w}(i) + (\overline{w}(i)$ $\underline{w}(i))\xi_i$ for each i 2 I?

Theorem 3. *The ARMKP includes Σ_2^P-hard problems.*

Proof. Given any instance of KIP, say $(V, I, v, w, p, \Gamma, W)$, we define, in polynomial time, an instance of ARMKP, denoted as $(\hat{V}, \hat{I}, \hat{K}, \underline{\hat{w}}, \overline{\hat{w}}, \hat{p}, \hat{\Xi}, \hat{W})$, such that KIP with data $(V, I, v, w, p, \Gamma, W)$ has an answer no if and only the ARMKP with data $(\hat{V}, \hat{I}, \hat{K}, \underline{\hat{w}}, \overline{\hat{w}}, \hat{p}, \hat{\Xi}, \hat{W})$ has an answer yes. We define $\hat{V} = V$, $\hat{I} = I$, $\hat{K} = f\Box g$ and set $\hat{W}(\Box) = W$. We also let $\underline{\hat{w}} = w$, $\overline{\hat{w}} = W + \epsilon$ (with $\epsilon > 0$) and $\hat{p} = p$. Finally, we define $\hat{\Xi}$ as $f\xi \ 2 \ f0, 1g^{j\hat{I}j} : \sum_{i2\hat{I}} v(i)\xi_i \quad \Gamma g$.

Since $j\hat{K}j = 1$, there exists only one partition of the item set, i,e. $Q_1 = \hat{I}$; consequently, the target value for the first conjunction is $\hat{v}_1 = \hat{V}$. Assume now that the ARMKP instance has answer yes. Then, for all $\xi \ 2 \ \hat{\Xi}$, the KP instance induced by items' weights $w(i) = \underline{w}(i) + (\overline{w}(i) \quad \underline{w}(i))\xi_i$ has answer yes. In other words, there is no $\xi \ 2 \ f0, 1g^{j\hat{I}j}$ such that $\sum_{i2\hat{I}} v(i)\xi_i \quad \Gamma$ and for which KNAPSACK PROBLEM with data $(v_1, Q_1, w, p, W(k))$ has an answer no. This proves that there is no $S \quad \hat{I}$ such that $\sum_{i2S} v(i) \quad \Gamma$ and KNAPSACK PROBLEM with data (V, InS, w, p, W) has an answer no, i.e., that KIP with data $(V, I, v, w, p, \Gamma, W)$ has answer no.
Vice-versa, if ARMKP has answer no, there exists a vector $\xi \ 2 \ f0, 1g^{j\hat{I}j}$ for which the associated KP instance has answer no. Item set $S = fi \ 2 \ \hat{I} : \xi_i = 1g$ is a certificate for the KIP instance with data $(V, I, v, w, p, \Gamma, W)$ to have a positive answer.

□

5.4 Theoretical development

This section presents the main theoretical development of our work. We start by reformulating problems of type (5.7) as adjustable robust problems with objective uncertainty; then, we consider a relaxation introduced in Kämmerling and Kurtz [2020] for this class of problems, introduce a solution algorithm for the relaxation and discuss its computational complexity. In the next section, we will discuss how this relaxation can be embedded in a branch-and-bound algorithm to close the optimality gap.

5.4.1 Reformulation

We first linearize every product involving variables ξ_{ij} and y_j for some (i, j) in constraints (5.6). An exact reformulation of each product can be obtained by introducing continuous variables $z_{ij} = \xi_{ij} y_j$ and adding the following linear constraints to the definition of set $Y(\hat{\boldsymbol{x}}, \hat{\boldsymbol{\xi}})$.

$$z_{ij} \quad u_j \xi_{ij} \qquad\qquad i = 1, ..., m_Y, j = 1, ..., n_Y \tag{5.11}$$

$$z_{ij} \quad y_j \qquad\qquad i = 1, ..., m_Y, j = 1, ..., n_Y \tag{5.12}$$

$$z_{ij} \quad y_j \quad (1 \quad \xi_{ij})u_j \qquad\qquad i = 1, ..., m_Y, j = 1, ..., n_Y \tag{5.13}$$

$$z_{ij} \quad 0 \qquad\qquad i = 1, ..., m_Y, j = 1, ..., n_Y \tag{5.14}$$

Let us introduce, for all $\boldsymbol{x} \ 2 \ X$ and all $\xi \ 2 \ \Xi$, set $Z(\boldsymbol{x}, \boldsymbol{\xi})$ as the set of decisions $(\boldsymbol{y}, \boldsymbol{z}) \ 2 \ \mathsf{R}^{n_Y} \quad \mathsf{R}^{m_Y \ n_Y}$ fulfilling constraints (5.4), (5.5),(5.12), (5.13) and (5.14) as well as constraints (5.15) obtained by replacing each bilinear terms in (5.6).

$$\sum_{j=1}^{n_X} t_{ij} x_j + \sum_{j=1}^{n_Y} \left(\underline{h}_{ij} y_j + (\bar{h}_{ij} \quad \underline{h}_{ij}) z_{ij}\right) \quad f_i \qquad\qquad i = 1, ..., m_Y \tag{5.15}$$

Remark 14. *We enlight that constraints* (5.11) *are not part of the de nition of* $Z(\boldsymbol{x},\boldsymbol{\xi})$ *and that this can be done without changing the optimal objective value of the second stage.*

Proof. We show that, given any solution of the second-stage problem violating (5.11), one can build a solution respecting (5.11) with the same objective value. Let $\bar{i},\bar{j}$ be the indices of a violated constraint (5.11) and note that $z_{\bar{i}\bar{j}} > 0$. Since $z_{\bar{i}\bar{j}}$ $y_{\bar{j}}$ $u_{\bar{j}}$, violation of the constraint implies $\xi_{\bar{i}\bar{j}} = 0$. Therefore (5.13) reduces to $z_{\bar{i}\bar{j}}$ $y_{\bar{j}}$ $u_{\bar{j}}$, which remains satisfied by setting $z_{\bar{i}\bar{j}} = 0$. Note that, since $(\bar{h}_{\bar{i}\bar{j}}$ $\underline{h}_{\bar{i}\bar{j}})$ 0 (see Assumption M), setting $z_{\bar{i}\bar{j}} = 0$ satisfies constraint (5.15) as well, as it only reduces its left-hand-side. Finally, notice that $z_{\bar{i}\bar{j}}$ does not appear in the objective function, i.e., the two solutions have the same value. □

In turn, it is clear that problem (5.7) is equivalent to the following adjustable robust problem

$$\min_{\boldsymbol{x}\in X}\left\{\boldsymbol{c}^T\boldsymbol{x} + \max_{\boldsymbol{\xi}\in\Xi}\min_{(\boldsymbol{y},\boldsymbol{z})\in Z(\boldsymbol{x},\boldsymbol{\xi})}\boldsymbol{d}^T\boldsymbol{y}\right\} \tag{5.16}$$

where the linking constraints between $\boldsymbol{y}$, $\boldsymbol{z}$ and $\boldsymbol{\xi}$ are of simpler kind. In the next theorem, we turn problem (5.16) into an adjustable robust problem where the uncertainty is confined within the objective function. This result follows from a polyhedral analysis result and Lagrangian duality.

Theorem 4. *Problem* (5.7) *is equivalently solved by the following problem.*

$$\min_{\boldsymbol{x}\in X}\left\{\sum_{j=1}^{n_X} c_j x_j + \max_{\boldsymbol{\xi}\in\Xi,\boldsymbol{\lambda}}\ {}_{0}\min_{(\boldsymbol{y},\boldsymbol{z})\in Z^0(\boldsymbol{x})}\sum_{j=1}^{n_Y}\left(d_j y_j + \sum_{i=1}^{m_Y}\lambda_{ij}\xi_{ij}(z_{ij}\quad y_j)\right)\right\} \tag{5.17}$$

where $Z^0(\boldsymbol{x})$ *is de ned as* $Z(\boldsymbol{x},\boldsymbol{\xi})$ *after omitting constraints* (5.13).

Proof. As already observed, one can consider problem (5.16) instead of (5.7). Then, by linearity of the objective function, condition "$(\boldsymbol{y},\boldsymbol{z})$ 2 $Z(\boldsymbol{x},\boldsymbol{\xi})$" can equivalently be replaced by "$(\boldsymbol{y},\boldsymbol{z})$ 2 conv$(Z(\boldsymbol{x},\boldsymbol{\xi}))$". Moreover, it holds that, for all $\boldsymbol{\xi}$ 2 Ξ, conv$(Z(\boldsymbol{x},\boldsymbol{\xi}))$ = conv$(Z^0(\boldsymbol{x}))$ \ $f(\boldsymbol{y},\boldsymbol{z})$: (5.13)g (see Theorem 14 in appendix). Thus, using a Dantzig-Wolfe reformulation of conv$(Z^0(\boldsymbol{x}))$, one can see the inner minimization problem as the solution of an LP for which strong Lagrangian duality holds, provided that the primal problem is feasible. Since we have assumed that $Y(\boldsymbol{x},\boldsymbol{\xi})$ 6= ; for all $\boldsymbol{x}$ 2 X and $\boldsymbol{\xi}$ 2 Ξ, strong duality holds. The partial Lagrangian dual is given as follows, where $\boldsymbol{\lambda}$ are the dual variables associated to the interdiction constraints (5.13).

$$\max_{\boldsymbol{\lambda}\ 0}\ \min_{(\boldsymbol{y},\boldsymbol{z})\in Z^0(\boldsymbol{x})}\left\{\sum_{j=1}^{n_Y} d_j y_j + \sum_{i=1}^{m_Y}\sum_{j=1}^{n_Y}\lambda_{ij}((1\quad\xi_{ij})u_j + z_{ij}\quad y_j)\right\} \tag{5.18}$$

By splitting the terms, we obtain:

$$\max_{\boldsymbol{\lambda}\ 0}\ \min_{(\boldsymbol{y},\boldsymbol{z})\in Z^0(\boldsymbol{x})}\left\{\sum_{j=1}^{n_Y} d_j y_j + \sum_{i=1}^{n_Y}\left(\sum_{j:\xi_{ij}=0}\lambda_{ij}(u_j + z_{ij}\quad y_j) + \sum_{j:\xi_{ij}=1}\lambda_{ij}(z_{ij}\quad y_j)\right)\right\} \tag{5.19}$$

We now argue that $\lambda_{ij} = 0$ is optimal whenever $\xi_{ij} = 0$. In this case, λ_{ij} is multiplied by $z_{ij} + u_j$ y_j, which is nonnegative as z_{ij} 0 and y_j u_j. Given that λ_{ij} 0, an optimal choice for the outer maximization problem is $\lambda_{ij} = 0$. □

The reformulation introduced in Theorem 4 is conceptually simpler than problem (5.7) as

uncertainty interferes within the objective function only. In other words, as shown by the following example, the feasible space does not depend on $\boldsymbol{\xi}$.

Example 9. *Let us consider a numerical example, in which the here-now-variables have been fixed and the resulting inner optimization problem is as follows:*

$$\max_{\boldsymbol{\xi} \in [0,1]^2 : \boldsymbol{e}^T \boldsymbol{\xi} \leq 1} \begin{array}{ll} \min & y_1 + y_2 \\ \text{s.t.} & (1+\xi_1) y_1 + (1+2\xi_2) y_2 \geq 3 \\ & y_1, y_2 \in \{0,1\} \end{array} \tag{5.20}$$

It is easily seen (e.g., by enumeration) that the optimal objective value is -1*, obtained by choosing* $\xi_1 = 0$ *and* $\xi_2 = 1$*. A direct application of Theorem 4 yields the following reformulation.*

$$\max_{\boldsymbol{\xi} \in [0,1]^2, \boldsymbol{\lambda} \in \mathbb{R}^2 : \boldsymbol{e}^T \boldsymbol{\xi} \leq 1} \begin{array}{ll} \min & -y_1 - y_2 + \xi_1 \lambda_1 (z_1 - y_1) + \xi_2 \lambda_2 (z_2 - y_2) \\ \text{s.t.} & y_1 + z_1 + y_2 + 2z_2 \geq 3 \\ & z_1 \geq y_1 \\ & z_2 \geq y_2 \\ & y_1, y_2 \in \{0,1\} \\ & z_1, z_2 \geq 0 \end{array} \tag{5.21}$$

Note that, for any $\boldsymbol{\xi}$*, the feasible space does not change. Thus,* $(y_1, y_2, z_1, z_2) = (1, 1, 0, 0)$ *is always feasible. However, this solution is optimal only when* $\xi_1 = \xi_2 = 0$ *and the objective is* $-y_1 - y_2 = -2$*. Yet, if we consider* $\xi_1 = 0$ *and* $\xi_2 = 1$*, the objective is now* $-y_1 - y_2 + \lambda_2(z_2 - y_2)$*. Since* $\lambda_2 \leq 0$ *and* $z_2 - y_2 \geq 0$*, the inner minimization problem will "tend to" minimize the distance between* z_2 *and* y_2*. With a negative enough value for* λ_2*, such a penalization will eventually force* $y_2 = z_2$ *since any feasible solution with* $y_2 \neq z_2$ *will be dominated. Observe that, for this example,* $\lambda_2 = -1$ *is enough. Similarly. when* $\xi_1 = 1$ *and* $\xi_2 = 0$*, one can show that* $\lambda_1 = -1$ *is enough. Theorem 5 shows how to compute sufficiently negative* λ *values for a general class of problems.*

We now discuss a possible way to fix variables λ_{ij} to an optimal value in the reformulation, so as to omit bilinear terms $\lambda_{ij}\xi_{ij}$.

Corrolary 5. *Let* $\lambda_{ij}(\boldsymbol{\xi})$ *be an optimal solution associated to a given* $\boldsymbol{\xi} \in \Xi$ *for problem* (5.18) *and let* $\underline{\lambda}_{ij}$ *be such that* $\underline{\lambda}_{ij} \leq \lambda_{ij}(\boldsymbol{\xi}) \leq 0$ *for all* $\boldsymbol{\xi} \in \Xi$*. Then, problem* (5.7) *can be rewritten as follows:*

$$\min_{\boldsymbol{x} \in X} \left\{ \sum_{j=1}^{n_X} c_j x_j + \max_{\boldsymbol{\xi} \in \Xi} \min_{(\boldsymbol{y}, \boldsymbol{z}) \in Z^0(\boldsymbol{x})} \sum_{j=1}^{n_Y} \left(d_j y_j + \sum_{i=1}^{m_Y} \underline{\lambda}_{ij} \xi_{ij} (z_{ij} - y_j) \right) \right\} \tag{5.22}$$

Proof. By definition of $\lambda_{ij}(\boldsymbol{\xi})$, problem (5.17) is equivalent to

$$\min_{\boldsymbol{x} \in X} \left\{ \sum_{j=1}^{n_X} c_j x_j + \max_{\boldsymbol{\xi} \in \Xi} \min_{(\boldsymbol{y}, \boldsymbol{z}) \in Z^0(\boldsymbol{x})} \sum_{j=1}^{n_Y} \left(d_j y_j + \sum_{i=1}^{m_Y} \lambda_{ij}(\boldsymbol{\xi}) \xi_{ij} (z_{ij} - y_j) \right) \right\} \tag{5.23}$$

By optimality, for a given $\boldsymbol{\xi}^0 \in \Xi$, since $\lambda_{ij}(\boldsymbol{\xi}^0) \leq 0$ and $z_{ij} \geq y_j$, any value $\underline{\lambda}_{ij} \leq \lambda_{ij}(\boldsymbol{\xi}^0)$ is also an optimal solution for $\boldsymbol{\xi}^0$. This achieves the proof. □

Corollary 5 therefore eliminates the need for variables $\boldsymbol{\lambda}$ by replacing them with a fixed and

exact violation penalization in the objective. However, it does not provide a practical value for $\underline{\lambda}_{ij}$ which would not be problem-specific. In the next theorem, we give such a value for a large class of problems.

Lemma 5. *In Corollary 5, assume that* $d_j \geq 0$ *and* $\underline{h}_{ij} \geq 0$ $(i = 1, ..., m_Y$ *and* $j = 1, ..., n_Y)$*. Then, one can safely use* $\underline{\lambda}_{ij} = d_j$ *for all* $i = 1, ..., m_Y$ *and* $j = 1, ..., n_Y$*, provided that the requirement* $z_{ij} \in \{0, 1\}$ *is added to the definition of* $Z^0(\boldsymbol{x})$ *for all* $i = 1, ..., m_Y$ *and all* $j \in \{1, ..., n_Y\}$ *such that* y_j *is a binary variable.*

Proof. Note that the inner maximization term of (5.17) can be reformulated as follows:

$$\max \left\{ \theta : \begin{array}{ll} \theta \leq \sum_{j=1}^{n_Y} \left(d_j y_j + \sum_{i=1}^{m_Y} \lambda_{ij} \xi_{ij} (z_{ij} - y_j) \right) & \forall (\boldsymbol{y}, \boldsymbol{z}) \in Z^0(\boldsymbol{x}) \\ \lambda_{ij} \geq 0 & i = 1, ..., m_Y, j = 1, ..., n_Y \\ \theta \in \mathbb{R} & \end{array} \right\} \tag{5.24}$$

Let $(\theta^*, \boldsymbol{\lambda}^*)$ be an optimal solution of (5.24). By strong duality, the following holds:

$$\theta^* = \min_{(\boldsymbol{y}, \boldsymbol{z}) \in Z(\boldsymbol{x}, \boldsymbol{\xi})} \boldsymbol{d}^T \boldsymbol{y} \tag{5.25}$$

Our goal is to derive a family of valid inequalities for (5.24), depending on points in $Z^0(\boldsymbol{x})$ but not on Λ, that bound θ^* from above.
To this end, let $(\boldsymbol{y}^0, \boldsymbol{z}^0)$ be any point in $Z^0(\boldsymbol{x})$, and define a point $(\hat{\boldsymbol{y}}, \hat{\boldsymbol{z}})$ as follows: for all $j \in \{1, ..., n_Y\}$,

- If $\xi_{ij} = 0$ for every $i = 1, ..., m_Y$, set $\hat{y}_j = y^0_j$ and $\hat{z}_{ij} = 0$ for all $i = 1, ..., m_Y$.
- Otherwise, there exists at least one index $i \in \{1, ..., m_Y\}$ such that $\xi_{ij} = 1$.
  Let $\bar{i}_j \in \arg\min_{i \in \{1, ..., m_Y\}} \{z^0_{ij} : \xi_{ij} = 1\}$ and set $\hat{y}_j = z^0_{\bar{i}_j j}$ and $\hat{z}_{ij} = \xi_{ij} \hat{y}_j$ for all $i = 1, ..., m_Y$.

Note that, in both cases, we have $\hat{y}_j \geq y^0_j$ for all j. This is straightforward in the first case, and is enforced by (5.12) in the latter as $\hat{y}_j = z^0_{\bar{i}_j j}$. In addition, we have $\hat{z}_{ij} \geq z^0_{ij}$ for all i and j. Now, since $\underline{h}_{ij} \geq 0$, $(\hat{\boldsymbol{y}}, \hat{\boldsymbol{z}}) \geq (\boldsymbol{y}^0, \boldsymbol{z}^0)$, and $\hat{y}_j = \hat{z}_{ij}$ when $\xi_{ij} = 1$, we can conclude that $(\hat{\boldsymbol{y}}, \hat{\boldsymbol{z}}) \in Z(\boldsymbol{x}, \boldsymbol{\xi})$. From (5.25) it follows that, given any feasible solution (θ, Λ) to (5.24), we have:

$$\theta \geq \sum_{j=1}^{n_Y} d_j \hat{y}_j = \sum_{j \in N} d_j \hat{y}_j + \sum_{j \in \{1, ..., n_Y\} \setminus N} d_j \hat{y}_j \tag{5.26}$$

where N denotes the set of indices of variables for which $\xi_{ij} = 0$ for all $i = 1, ..., n_Y$. Observe that, for each $j \in N$, we have $\hat{y}_j = y^0_j$, hence

$$\sum_{j \in N} d_j \hat{y}_j = \sum_{j \in N} d_j y^0_j = \sum_{j \in N} \left(d_j y^0_j + \sum_{i=1}^{m_Y} d_j \xi_{ij} (z^0_{ij} - y^0_j) \right). \tag{5.27}$$

As to the remaining variables, we have

$$\sum_{j \in \{1, ..., n_Y\} \setminus N} d_j \hat{y}_j = \sum_{j \in \{1, ..., n_Y\} \setminus N} d_j z^0_{\bar{i}_j j} = \sum_{j \in \{1, ..., n_Y\} \setminus N} \left(d_j y^0_j + d_j \xi_{\bar{i}_j j} (z^0_{\bar{i}_j j} - y^0_j) \right),$$

where the last equality holds as $\xi_{\bar{i}_j j} = 1$. Since, for each i and j, $d_j(z^0_{ij} \quad y^0_j) \quad 0$, we have,

$$\sum_{j \in \{1,\ldots,n_Y\}} d_j \hat{y}_j \quad \sum_{j \in \{1,\ldots,n_Y\}} \left(d_j y^0_j + d_j \xi_{\bar{i}_j j}(z^0_{\bar{i}_j j} \quad y^0_j) + \sum_{i \neq \bar{i}_j} d_j \xi_{ij}(z^0_{ij} \quad y^0_j) \right) =$$

$$\sum_{j \in \{1,\ldots,n_Y\}} \left(d_j y^0_j + \sum_{i=1}^{m_Y} d_j \xi_{ij}(z^0_{ij} \quad y^0_j) \right) \tag{5.28}$$

By combining (5.26), (5.27), and (5.28), we obtain

$$\theta \quad \sum_{j \in N} \left(d_j y^0_j + \sum_{i=1}^{m_Y} d_j \xi_{ij}(z^0_{ij} \quad y^0_j) \right) + \sum_{j \in \{1,\ldots,n_Y\}} \left(d_j y^0_j + \sum_{i=1}^{m_Y} d_j \xi_{ij}(z^0_{ij} \quad y^0_j) \right) =$$

$$\sum_{j=1}^{n_Y} \left(d_j y^0_j + \sum_{i=1}^{m_Y} d_j \xi_{ij}(z^0_{ij} \quad y^0_j) \right) \tag{5.29}$$

Given the arbitrary choice of point $(\boldsymbol{y}^0, \boldsymbol{z}^0) \in Z^0(\boldsymbol{x})$, the following family of inequalities are valid for (5.24):

$$\theta \quad \sum_{j=1}^{n_Y} \left(d_j y_j + \sum_{i=1}^{m_Y} d_j \xi_{ij}(z_{ij} \quad y_j) \right) \quad \forall (\boldsymbol{y}, \boldsymbol{z}) \in Z^0(\boldsymbol{x}) \tag{5.30}$$

We know from Corollary 5 than all negative enough Λ values are optimal. Any inequality of (5.24) would be dominated by these cuts when $\lambda_{ij} \quad d_j$, making $\lambda_{ij} = d_j$ a safe choice. □

Example 10. *We continue the example based on the problem introduced in Example 9 to discuss the interpretation of Theorem 5. As anticipated, a large enough value for λ_1 and λ_2 is* $\quad 1$. *The objective function now becomes* $\quad y_1 \quad y_2 \quad \xi_1(z_1 \quad y_1) \quad \xi_2(z_2 \quad y_2)$ *which, once re-organized, is equivalently written as follows.*

$$y_1(1 \quad \xi_1) \quad \xi_1 z_1 \quad y_2(1 \quad \xi_2) \quad \xi_2 z_2 \tag{5.31}$$

A very clear interpretation is now at hand: when $\xi_1 = 0$, then the contribution of (y_1, z_1) to the objective is $\quad y_1$, *whereas this term reduces to* $\quad z_1$ *if $\xi_1 = 1$. In other words, setting $\xi_1 = 1$ and $\xi_2 = 0$ forces $z_1 = y_1$. More in general, for a given $\boldsymbol{\xi} \in \Xi$, setting $\xi_{ij} = 1$ induces a contribution $\bar{h}_{ij} y_j$ in the left-hand-side of the i-th constraint* (5.15).

Theorem 5 achieves the ultimate goal of reformulating problem (5.7) as an adjustable robust optimization problem with objective uncertainty. We emphasize that this result generalizes the work of Fischetti et al. [2019] on bi-level interdiction games in two directions: (i) it extends their results (in particular, Theorem 2 and 3) to two-stage robust problems; and (ii) it provides a dual interpretation of valid cost penalization.

5.4.2 Relaxation

In the previous section, we have reformulated problem (5.7) so as to obtain an objective-uncertain ARO. This class of problems has been studied, among others, in Kämmerling and Kurtz [2020], Arslan and Detienne [2021] and Detienne et al. [2021]; in many of these works, only the case in

which here-and-now variables are binary and the uncertainty set convex is considered. We now make a step further in the analysis of uncertain adjustable robust problems, and consider the more general case with mixed-integer here-and-now decisions, and in which the uncertainty set is binary.

To ease our presentation, we first assume that problem (5.7) is feasible. Note that this assumption is easily enforced in practice by adding decision variables with high costs (similarly to Phase I in the Simplex algorithm). Solving ARO problems without this assumption will be discussed in Section 5.4.4.

Assumption N (Feasibility)**.** *Problem* (5.7) *is feasible.*

Theorem 5. *Let* v *be the optimal objective value of problem* (5.7) *and let* v_R *be the optimal objective value of the following problem*

$$\max \theta \tag{5.32}$$

$$\text{s. t. } \theta \quad \sum_{j=1}^{n_X} c_j \hat{x}_j + \sum_{j=1}^{n_Y} \left(d_j \hat{y}_j + \sum_{i=1}^{m_Y} \underline{\lambda}_{ij} \xi_{ij} (\hat{z}_{ij} \quad \hat{y}_j) \right) \qquad 8(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}, \hat{\boldsymbol{z}}) \; 2 \; W \tag{5.33}$$

$$\boldsymbol{\xi} \; 2 \; \Xi, \tag{5.34}$$

where W *denotes the set of extreme points of the convex hull of* $f(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) : \boldsymbol{x} \; 2 \; X, (\boldsymbol{y}, \boldsymbol{z}) \; 2 \; Z^0(\boldsymbol{x})g$*. Then,* $v \quad v_R$*. Moreover, if all active constraints of type* (5.33) *are built upon the same here-and-now decision, say* $\bar{\boldsymbol{x}}$*, then* $v \; = v_R$ *and* $\bar{\boldsymbol{x}}$ *solves problem* (5.7)*.*

Proof. The first part of the theorem comes from the min-max inequality; indeed, for any function $f : A \quad B \; ! \; \mathsf{R}$, it holds that $\sup_{\boldsymbol{a} 2 A} \inf_{\boldsymbol{b} 2 B} f(\boldsymbol{a}, \boldsymbol{b}) \quad \inf_{\boldsymbol{b} 2 B} \sup_{\boldsymbol{a} 2 A} f(\boldsymbol{a}, \boldsymbol{b})$.

As to the second part, assume that all the active cuts are built upon the same here-and-now variable, say $\bar{\boldsymbol{x}}$ (note that, by Assumption N, $W \; 6= \; ;$). Then,

$$v_R = \max_{\boldsymbol{\xi} 2 \Xi} \min_{(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) 2 f \bar{\boldsymbol{x}} g \quad Z^0(\bar{\boldsymbol{x}})} \left\{ \sum_{j=1}^{n_X} c_j \bar{x}_j + \sum_{j=1}^{n_Y} \left(d_j y_j + \sum_{i=1}^{m_Y} \underline{\lambda}_{ij} \xi_{ij} (z_{ij} \quad y_j) \right) \right\} \tag{5.35}$$

$$= \min_{\boldsymbol{x} = \bar{\boldsymbol{x}}} \max_{\boldsymbol{\xi} 2 \Xi} \min_{(\boldsymbol{y}, \boldsymbol{z}) 2 Z^0(\bar{\boldsymbol{x}})} \left\{ \sum_{j=1}^{n_X} c_j \bar{x}_j + \sum_{j=1}^{n_Y} \left(d_j y_j + \sum_{i=1}^{m_Y} \underline{\lambda}_{ij} \xi_{ij} (z_{ij} \quad y_j) \right) \right\} \tag{5.36}$$

$$v \tag{5.37}$$

□

Though the relaxation introduced in Theorem 5 consists in a monolithic Mixed Integer Linear Program (MILP), it contains an exponential number of cuts of type (5.33). For the sake of simplicity, let us introduce the following function:

$$\Pi(\boldsymbol{x}, \boldsymbol{\xi}, \boldsymbol{y}, \boldsymbol{z}) := \sum_{j=1}^{n_X} c_j x_j + \sum_{j=1}^{n_Y} \left(d_j y_j + \sum_{i=1}^{m_Y} \underline{\lambda}_{ij} \xi_{ij} (z_{ij} \quad y_j) \right) \tag{5.38}$$

The procedure described in Algorithm 3 is a cut-generation approach for solving problem (5.32)-(5.34). The finite convergence of the obtained algorithm is well-established. Algorithm 3 includes

an initialization phase assuming that (at least) one point $(\boldsymbol{x}^0, \boldsymbol{y}^0, \boldsymbol{z}^0)$ such that $\boldsymbol{x}^0 \in X$ and $(\boldsymbol{y}^0, \boldsymbol{z}^0) \in Z^0(\boldsymbol{x}^0)$ is available. For example, in the common case in which $\mathbf{0} \in \Xi$, such a point may be found by solving the following problem, corresponding to the deterministic version of the original problem:

$$(\boldsymbol{x}^0, \boldsymbol{y}^0, \boldsymbol{z}^0) \in \operatorname{argmin}\{\Pi(\boldsymbol{x}, \mathbf{0}, \boldsymbol{y}, \boldsymbol{z}) : \boldsymbol{x} \in X, (\boldsymbol{y}, \boldsymbol{z}) \in Z^0(\boldsymbol{x})\} \tag{5.39}$$

Algorithm 3: Cut-generation algorithm

Initialize $\hat{W}$ with some point $(\boldsymbol{x}^0, \boldsymbol{y}^0, \boldsymbol{z}^0)$ with $\boldsymbol{x}^0 \in X$ and $(\boldsymbol{y}^0, \boldsymbol{z}^0) \in Z^0(\boldsymbol{x}^0)$.
repeat
  $(\theta, \boldsymbol{\xi}) \in \operatorname{argmax}\left\{\theta : (\theta, \boldsymbol{\xi}) \in \mathbb{R} \times \Xi \times \theta \leq \Pi(\hat{\boldsymbol{x}}, \boldsymbol{\xi}, \hat{\boldsymbol{y}}, \hat{\boldsymbol{z}}) \quad \forall (\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}, \hat{\boldsymbol{z}}) \in \hat{W}\right\}$
  $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) \in \operatorname{argmin}\{\Pi(\boldsymbol{x}, \boldsymbol{\xi}, \boldsymbol{y}, \boldsymbol{z}) : \boldsymbol{x} \in X, (\boldsymbol{y}, \boldsymbol{z}) \in Z^0(\boldsymbol{x})\}$
  $\hat{W} \leftarrow \hat{W} \cup \{(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})\}$
until $\theta \leq \Pi(\boldsymbol{x}, \boldsymbol{\xi}, \boldsymbol{y}, \boldsymbol{z})$
stop, and $v_R = \theta$.

5.4.3 Solving the separation problem

In this subsection, we discuss the complexity of the separation problem of Algorithm 3. This separation problem is recalled here: for a fixed $\bar{\boldsymbol{\xi}} \in \Xi$, this problem reads:

$$\min \sum_{j=1}^{n_X} c_j x_j + \sum_{j=1}^{n_Y} \left(d_j y_j + \sum_{i=1}^{m_Y} \underline{\lambda}_{ij} \bar{\xi}_{ij} (z_{ij} - y_j) \right) \tag{5.40}$$

$$\text{s.t. } \boldsymbol{x} \in X, (\boldsymbol{y}, \boldsymbol{z}) \in Z^0(\boldsymbol{x}) \tag{5.41}$$

We make the following remark on solving the separation problem.

Remark 15 (Relation with the deterministic problem)**.** *The separation problem can be solved by any oracle designed for solving the deterministic problem* (5.1)-(5.5) *with appropriate input values.*

Proof. Reading the proofs of Corollary 5 and Theorem 4 in the reversed order, the following holds.

$$\min_{\boldsymbol{x} \in X, (\boldsymbol{y}, \boldsymbol{z}) \in Z^0(\boldsymbol{x})} \Pi(\boldsymbol{x}, \boldsymbol{\xi}, \boldsymbol{y}, \boldsymbol{z}) \tag{5.42}$$

$$= \min_{\boldsymbol{x} \in X} \left\{ \boldsymbol{c}^T \boldsymbol{x} + \min_{(\boldsymbol{y}, \boldsymbol{z}) \in Z^0(\boldsymbol{x})} \boldsymbol{d}^T \boldsymbol{y} + \sum_{j=1}^{n_Y} \sum_{i=1}^{m_Y} \underline{\lambda}_{ij} \xi_{ij} (z_{ij} - y_j) \right\} \tag{5.43}$$

$$= \min_{\boldsymbol{x} \in X} \left\{ \boldsymbol{c}^T \boldsymbol{x} + \min_{(\boldsymbol{y}, \boldsymbol{z}) \in \operatorname{conv}(Z^0(\boldsymbol{x}))} \boldsymbol{d}^T \boldsymbol{y} + \sum_{j=1}^{n_Y} \sum_{i=1}^{m_Y} \underline{\lambda}_{ij} \xi_{ij} (z_{ij} - y_j) \right\} \tag{5.44}$$

$$= \min_{\boldsymbol{x} \in X} \left\{ \boldsymbol{c}^T \boldsymbol{x} + \max_{\boldsymbol{\lambda} \geq \mathbf{0}} \min_{(\boldsymbol{y}, \boldsymbol{z}) \in \operatorname{conv}(Z^0(\boldsymbol{x}))} \boldsymbol{d}^T \boldsymbol{y} + \sum_{j=1}^{n_Y} \sum_{i=1}^{m_Y} \lambda_{ij} ((1 - \xi_{ij}) u_j + z_{ij} - y_j) \right\} \tag{5.45}$$

$$= \min_{\boldsymbol{x} \in X} \left\{ \boldsymbol{c}^T \boldsymbol{x} + \min_{(\boldsymbol{y}, \boldsymbol{z}) \in Z(\boldsymbol{x}, \bar{\boldsymbol{\xi}})} \boldsymbol{d}^T \boldsymbol{y} \right\} \tag{5.46}$$

$$= \min_{\boldsymbol{x} \in X, \boldsymbol{y} \in Y(\boldsymbol{x}, \bar{\boldsymbol{\xi}})} \boldsymbol{c}^T \boldsymbol{x} + \boldsymbol{d}^T \boldsymbol{y} \tag{5.47}$$

Now, given an optimal solution $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ of the deterministic problem arising when $\boldsymbol{\xi} = \bar{\boldsymbol{\xi}}$, an optimal solution to the separation problem is given by $(\boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{z}^*)$, where $\boldsymbol{z}^*$ is defined as $z^*_{ij} = \bar{\xi}_{ij} y^*_j$. □

Remark 16. *The separation problem and the deterministic problem* (5.1)-(5.5) *belong to the same complexity class.*

Proof. We already have that any instance of the separation problem can be solved using an oracle for the deterministic problem. We also show that any instance of the deterministic problem can be solved by any oracle solving the separation problem. Indeed, using $\bar{\boldsymbol{\xi}} = \mathbf{0}$, problem (5.40)-(5.41) reduces to the deterministic problem since variable $\boldsymbol{z}$ becomes useless and can be removed from the model. Indeed, $\boldsymbol{z}$ comes in -inequality constraints with positive coefficients and have no contribution to the objective function. Thus, solutions such that $\boldsymbol{z} > \mathbf{0}$ are dominated by those solutions with $\boldsymbol{z} = 0$. □

In the next remark, we also provide a simple way to generate lifted cuts we are shown to be stronger than those provided by Algorithm 3. These cuts counterbalance the possible degeneracy of the separation problem.

Remark 17 (Lifted cuts)**.** *Assume* $\boldsymbol{d}$ $\mathbf{0}$ *and consider an optimal solution* $(\boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{z}^*)$ *of the separation problem arising from* $\bar{\boldsymbol{\xi}} \in \Xi$*. Then,*

$$\forall \boldsymbol{\xi} \in \Xi, \quad \Pi(\boldsymbol{x}^*, \boldsymbol{\xi}, \boldsymbol{y}^*, \boldsymbol{z}^+) \quad \Pi(\boldsymbol{x}^*, \boldsymbol{\xi}, \boldsymbol{y}^*, \boldsymbol{z}^*) \tag{5.48}$$

i.e., cuts built with $\boldsymbol{z}^+$ *dominate those built with* $\boldsymbol{z}^*$*, where* $\boldsymbol{z}^+$ *is an optimal solution of the following problem:*

$$\min \sum_{i=1}^{m_Y} \sum_{j=1}^{n_Y} \underline{\lambda}_{ij} z_{ij} \tag{5.49}$$

$$\text{s. t. } (\boldsymbol{y}^*, \boldsymbol{z}) \in Z^0(\boldsymbol{x}^*) \tag{5.50}$$

$$z_{ij} = z^*_{ij} \qquad \forall (i,j) : \bar{\xi}_{ij} = 1 \tag{5.51}$$

$$z_{ij} \quad z^*_{ij} \qquad \forall (i,j) : \bar{\xi}_{ij} = 0 \tag{5.52}$$

Proof. Firstly, note that $\boldsymbol{z}^*$ is feasible for the lifting problem. And, it is easy to see that $(\boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{z}^+)$ is feasible for the separation problem. Moreover, notice, that, whenever $\bar{\xi}_{ij} = 1$, i.e., whenever variable z_{ij} appears in the objective function of the separation problem, we have $z_{ij} = z^*_{ij}$. Therefore, the two solutions, $(\boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{z}^*)$ and $(\boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{z}^+)$ have the same objective value with respect to $\bar{\boldsymbol{\xi}}$. Yet, we have that, for all (i,j) such that $\bar{\xi}_{ij} = 0$, z^+_{ij} z^*_{ij}. Therefore, we have that, for all $\boldsymbol{\xi} \in \Xi$, $\sum_{i=1}^{m_Y} \sum_{j=1}^{n_Y} \xi_{ij} \underline{\lambda}_{ij} (z^+_{ij}$ $y^*_j)$ $\sum_{i=1}^{m_Y} \sum_{j=1}^{n_Y} \xi_{ij} \underline{\lambda}_{ij} (z^*_{ij}$ $y^*_j)$ (since $\boldsymbol{\lambda}$ $\mathbf{0}$). □

Intuitively, the lifted cuts are expected to be better than the "degenerated" cuts arising from the natural solution of the separation problem since they contain more information about the possible recourse decisions which the decision maker can decide in the wait-and-see phase. Also notice that the same property holds for any sub-optimal solution for the lifting problem as well.

5.4.4 Dealing with infeasibility

When Assumption N holds, observe that the proposed relaxation is valid even for problems without complete recourse. Indeed, since the optimization order is changed in the relaxed problem (i.e., the scenario is decided first), it cannot be that a here-and-now decision without recourse decision is returned by the separation problem. As a matter of fact, Assumption N implies that there exists at least one here-and-now decision with a feasible recourse decision, for any given scenario.

Thus, the only feasibility check which needs to be done, in case Assumption N is not known to hold, is that, given scenario $\hat{\boldsymbol{\xi}} \in \Xi$ returned by the maximization problem, there indeed exists one $\boldsymbol{x} \in X$ such that $Y(\boldsymbol{x}, \hat{\boldsymbol{\xi}}) \neq \emptyset$. This condition is met if and only if the following optimization problem has an optimal objective value of zero:

$$\min \|\boldsymbol{s}\|_1 + \|\boldsymbol{u}\|_1 + \sum_{i=1}^{m_Y} \sum_{j=1}^{n_Y} \underline{\lambda}_{ij} \xi_{ij} (z_{ij} \quad y_j) \tag{5.53}$$

$$\text{s. t. } (\boldsymbol{x} + \boldsymbol{s}) \in X, (\boldsymbol{y} + \boldsymbol{u}) \in Z^0(\boldsymbol{x}) \tag{5.54}$$

$$\boldsymbol{s} \in \mathbb{R}^{n_X}, \boldsymbol{u} \in \mathbb{R}^{n_Y}. \tag{5.55}$$

If, to the contrary, the optimal objective value is strictly less than zero, then the following cut may be added,

$$0 \quad \|\boldsymbol{s}\| \|_1 + \|\boldsymbol{u}\| \|_1 + \sum_{i=1}^{m_Y} \sum_{j=1}^{n_Y} \underline{\lambda}_{ij} \xi_{ij} (z_{ij} \quad y_j), \tag{5.56}$$

where $(\boldsymbol{x}^*, \boldsymbol{s}^*, \boldsymbol{u}^*, \boldsymbol{y}^*, \boldsymbol{z}^*)$ denotes an optimal solution of (5.53)-(5.55).

5.5 A Branch-and-bound algorithm

In this section, we generalize the branch-and-bound scheme in Kämmerling and Kurtz [2020] to the case in which the here-and-now decisions are mixed-integer and the uncertainty set is binary. This extension requires non-trivial arguments for the convergence analysis of the resulting algorithm, as well as specific subroutines to account for the discrete nature of the uncertainty set. For the sake of simplicity, we present the algorithm for the case in which integer here-and-now variables are all binary, the extension to the general case being straightforward.

5.5.1 Statement of the procedure

For a given node q with local bounds noted $\boldsymbol{l}^q$ and $\boldsymbol{u}^q$, let $W^q := \{(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) : \boldsymbol{x} \in X \setminus [\boldsymbol{l}^q, \boldsymbol{u}^q], (\boldsymbol{y}, \boldsymbol{z}) \in Z^0(\boldsymbol{x})\}$ and let v_R^q denote the optimal objective value of (5.32)-(5.34) where W has been replaced by W^q. Additionally, we let H^q denote the set of active constraints of type (5.33) for the obtained problem (see Section 5.5.2) and $\boldsymbol{x}^{[h]}$ denote the here-and-now decision which was used to generate the h-th cut of type (5.33), for $h \in H^q$. Finally, let us introduce $\bar{\boldsymbol{x}}^q$ defined as follows:

$$\bar{\boldsymbol{x}}^q = \frac{1}{|H^q|} \sum_{h \in H^q} \boldsymbol{x}^{[h]}. \tag{5.57}$$

We denote by l_j^0 and u_j^0 the initial bounds for the here-and-now variables. Therefore, we have that $W^0 = W$, and the root node corresponds to computing v_R^0. We assume that our algorithm makes

use of a list L which contains all active nodes still to be explored in order to close the optimality gap. In addition, we assume that the best lower and upper bounds found, denoted by LB and UB, respectively, are dynamically updated during the execution of the algorithm.

At the beginning of the algorithm, we solve the root node (5.32)-(5.34) and set L $f0g$, LB v_R^0 and UB $+1$. Clearly, if $\bar{\boldsymbol{x}}^0 \in \{\boldsymbol{l}^0, \boldsymbol{u}^0\}$, the algorithm stops and the problem has been solved from the root node, i.e., v $= v_R^0$ and UB v_R^0. Otherwise, we iteratively perform the following steps:

Node selection A node q with minimal objective value is selected for branching, i.e.,

$$q \in \operatorname{argmin}\{v_R^p : p \in L\} \tag{5.58}$$

This operation may lead to an update of the best known lower bound, i.e., LB v_R^q.

Branching strategy Given a selected node q, we compute, for every here-and-now decision variable, a score $\theta_j^q \in [0,1]$ and select the variable with maximum score, prioritizing binary variables. In our implementation, we used the following score.

$$\theta_j^q = \begin{cases} 0 & \text{if } l_j^q = u_j^q \\ \min\{u_j \quad \bar{x}_j^q; \bar{x}_j^q \quad l_j^q\} / (u_j^q \quad l_j^q) & \text{otherwise} \end{cases} \tag{5.59}$$

Let j^0 be the index of the variable selected for branching; we create two new nodes q^1 and q^2 in which we set $l_j^{q^1} = l_j^{q^2} = l_j^q$ and $u_j^{q^1} = u_j^{q^2} = u_j^q$ for all $j \neq j^0$ and set $l_{j^0}^{q^1}, l_{j^0}^{q^2}, u_{j^0}^{q^1}$ and $u_{j^0}^{q^2}$ as follows: if j^0 corresponds to a binary variable, then $\bar{x}_{j^0}$ is fixed to 0 (resp. to 1) in q^1 (resp. in q^2), i.e., $l_{j^0}^{q^1} = u_{j^0}^{q^1} = 0$ and $l_{j^0}^{q^2} = u_{j^0}^{q^2} = 1$; otherwise, we set $l_{j^0}^{q^1} = l_{j^0}^q$, $u_{j^0}^{q^1} = \bar{x}_{j^0}^q$, $l_{j^0}^{q^2} = \bar{x}_{j^0}^q$, and $u_{j^0}^{q^2} = u_{j^0}^q$. The two created nodes are then solved in the following step.

Node solution For $q^0 \in \{q^1, q^2\}$, we compute $v_R^{q^0}$ and $\bar{\boldsymbol{x}}^{q^0}$. If "$\bar{\boldsymbol{x}}^{q^0} \in \left\{\boldsymbol{l}^{q^0}, \boldsymbol{u}^{q^0}\right\}$", a new incumbent solution has been found and we let UB $\min\{UB, v_R^{q^0}\}$. Otherwise, q^0 is added to the list of active nodes, i.e., L $L \cup \{q^0\}$.

Bounding For any active node $q^0 \in L$, if $v_R^{q^0}$ UB, then q^0 can be removed from L, i.e., L $L \setminus \{q^0\}$.

Stopping criteria If L is empty, the algorithm stops and $UB = v$.

In addition to the described steps of the branch-and-bound algorithm, note that it is also possible to compute feasible solutions thanks to the following proposition.

Proposition 5. *Let q be a given node and let us assume that $\bar{x}_j^q \in \{0,1\}$ for all j for which x_j is required to be binary. Then, a feasible solution for* (5.7) *can be computed by solving formulation* (5.32)-(5.34) *where W has been replaced by* $\operatorname{vertex}(\operatorname{conv}(\{\bar{\boldsymbol{x}}^q\} \quad Z^0(\bar{\boldsymbol{x}}^q)))$.

Proof. The proof is similar to that of Theorem 5. □

For every node q, when appropriate, we will denote by v_U^q the objective value obtained applying Proposition 5. Notice that, even when some binary here-and-now variables are fractional, one could still try to round fractional values to the closest integer and check if the resulting is feasible; in that case, v_U^q is a valid upper bound for (5.7).

5.5.2 Identifying active cuts

At each node of the branch-and-bound tree, a here-and-now decision is reconstructed from the solved relaxation by means of formula (5.57). This formula computes the "average" here-and-now decision among all active cuts, gathered in H^q. When the uncertainty set Ξ is a convex set, identifying the set of active constraints is easily done by checking the slack variables of each cut. To adapt the approach from Kämmerling and Kurtz [2020] to our binary uncertainty context, we should here discuss a procedure to identify active cuts when Ξ is binary. A key observation is that replacing H^q in (5.57) by the set of *generated* constraints, noted $\hat{H}^q$, does not change the overall validity of the procedure. Unfortunately, the number of generated constraints may be large, possibly leading to poor branching decisions and a large branch-and-bound tree. Indeed, the tightness condition from Theorem 5 may be identified only at a late instant. Similarly, the quality of the solution returned by the heuristic proposed in Proposition 5 may not be good enough to prune nodes at an early stage if the here-and-now decision $\bar{\boldsymbol{x}}^q$ is loosely computed. Instead, one could try to obtain the subset of generated constraints which prevent the objective value from growing, in the lower-bounding problem. To this end, let us introduce a given real $\varepsilon > 0$. Then, identifying the set of active constraints can be done by searching for an *Irreducible Infeasible Subsystem* (IIS) of the following infeasible model:

$$\max \theta \tag{5.60}$$

$$\text{s. t. } \theta \quad \sum_{j=1}^{n_X} c_j \hat{x}_j + \sum_{j=1}^{n_Y} \left(d_j \hat{y}_j + \sum_{i=1}^{m_Y} \lambda_{ij} \xi_{ij} (\hat{z}_{ij} \quad \hat{y}_j) \right) \qquad \forall (\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}, \hat{\boldsymbol{z}}) \in \hat{H}^q \tag{5.61}$$

$$\theta \quad v_R^q + \varepsilon \tag{5.62}$$

$$\boldsymbol{\xi} \in \Xi. \tag{5.63}$$

Depending on the selected tolerance ε, a subset $H^q \quad \hat{H}^q$ is identified as being part of the IIS and formula (5.57) can be applied to obtain a reconstructed here-and-now decision.

5.5.3 Convergence result

The enumerative scheme presented in the previous section solves problem (5.7) by iteratively partitioning the feasible set, possibly performing branching on a continuous variable. Clearly, if $X \quad \{0,1\}^{n_X}$, our algorithm finitely converges. We show that, otherwise, the algorithm still converges towards an optimal solution of (5.7) though, potentially, an infinite number of steps is required.

Theorem 6. *Let v denote the optimal solution value of problem (5.7). Then, the branch-and-bound algorithm either nitely terminates with $UB = v$ or enters an in nite sequence of nodes, say $p \in P$, for which $\{\bar{\boldsymbol{x}}^p\}_{p \in P} \to \boldsymbol{x}$, where $\boldsymbol{x}$ is an optimal solution of the problem.*

Proof. We focus on the case in which the algorithm enters an infinite sequence of nodes. Since our branching strategy prioritizes binary variables over continuous ones, we may assume that all variables of the former type are fixed in the sequence. Moreover, an infinite sequence of nodes implies the existence of an infinite subsequence of nodes belonging to the same branch; we denote this subsequence by P and its generic node by p. We will denote by $[\boldsymbol{l}^p, \boldsymbol{u}^p]$ the local bounds of

node p, by $(\theta^p\ ,\boldsymbol{\xi}^p\ )$ the optimal solution of problem (5.32)-(5.34), and by $\bar{\boldsymbol{x}}^p$ the here-and-now decision computed according to (5.57).
Since branching always reduces the domain of the here-and-now variables, we have $[\boldsymbol{l}^{p+1},\boldsymbol{u}^{p+1}]$ $[\boldsymbol{l}^p,\boldsymbol{u}^p]$. Thus, there exists a subsequence of P for which $[\boldsymbol{l}^p,\boldsymbol{u}^p]$ converges to $[\boldsymbol{l}\ ,\boldsymbol{u}\ ]$. In addition, there exists a subsequence for which $\boldsymbol{\xi}^p$ converges to $\boldsymbol{\xi}\ $, as $\boldsymbol{\xi}$ variables are binary and the number of their feasbile combinations is finite.

Since our node selection strategy always picks the node with lowest bound, we have that θ^p is a lower bound on $v\ $. In addition, branching operation implies that $\theta^{(p+1)}\quad\theta^p\ $, and thus θ^p converges to $\theta\ $. Finally, $\bar{\boldsymbol{x}}^p\ \ 2\ [\boldsymbol{l}^p,\boldsymbol{u}^p]\quad[\boldsymbol{l}^0,\boldsymbol{u}^0]$, hence there exists a subsequence for which $\bar{\boldsymbol{x}}^p$ converges to a solution $\bar{\boldsymbol{x}}\ $ that belongs to $[\boldsymbol{l}\ ,\boldsymbol{u}\ ]$; otherwise there would exist a node p for which $\bar{\boldsymbol{x}}^p\ \ \not{2}\ [\boldsymbol{l}^p,\boldsymbol{u}^p]$. In addition, $\bar{\boldsymbol{x}}\ \ 2\ X$ as X is a compact set.

We now show that $\bar{\boldsymbol{x}}\ \ 2\ \mathrm{vert}([\boldsymbol{l}\ ,\boldsymbol{u}\ ])$. Indeed, let $j\ 2\ \{1,...,n_X\}$ be a here-and-now variable index which is infinitely branched on, and consider the subsequence $P^0\quad P$ such that, for all $p\ 2\ P^0$, $u_j^{p+1}=x_j^p\ $. If P^0 is infinite, then $\{u_j^p\}_{p2P^0}\ !\ x_j$ since $\{x_j^p\}_{p2P^0}\ !\ x_j$ as $P^0\quad P$. If instead P^0 is finite, then the subsequence $P^{00}\quad P$ such that, for all $p\ 2\ P^{00}$, $l_j^{p+1}=x_j^p$ is infinite and, for the same reason, we have $\{l_j^p\}_{p2P^{00}}\ !\ x_j$.

Thus, we have that $\bar{\boldsymbol{x}}\ \ 2\ \mathrm{vert}([\boldsymbol{l},\boldsymbol{u}])$ and, by Theorem 5, its cost is $\theta\ $. In addition, $\bar{\boldsymbol{x}}$ is feasible for (5.7) and thus $\theta\quad v\ $. Since $\theta^p\quad v$ for all $p\ 2\ P$, this is true for θ as well, and hence θ coincides with the optimal solution value $v\ $.

□

5.6 Computational experiments

In this section, we study the ARMKP introduced in Section 5.3. We are given n items, the j-th characterized by a nominal weight $\bar{w}_j$ and by a profit p_j, and K identical knapsacks, each having a given capacity W. We assume that the weight of each item cannot be known here and now, i.e., when the assignment of an item to a knapsack is decided. Rather, the actual weight of each item j may take only two possible values, namely, $\underline{w}_j$ and $\bar{w}_j=\underline{w}_j+\tilde{w}_j$, with $\tilde{w}_j>0$.

In a first stage, every item has to be assigned to exactly one knapsack. The here-and-now feasible space X is therefore modeled by means of binary variables x_{jk}, defined for each item $j\ 2\ \{1,...,n\}$ and knapsack $k\ 2\ \{1,...,K\}$. Each variable takes value 1 iff item j is assigned to knapsack k. More formally, X is deined as follows

$$X=\left\{\boldsymbol{x}\ 2\ \{0,1\}^{n\quad K}:\sum_{k=1}^{K}x_{jk}=1,j=1,...,n\right\} \tag{5.64}$$

As to uncertainty, we assume that up to Γ ($\Gamma\ 2\ \mathbb{N}\cap\{0\}$) items may have a largest weight (i.e., $\bar{w}_j$) than their nominal weight (i.e., $\underline{w}_j$). We therefore consider the following uncertainty set,

$$\Xi=\left\{\boldsymbol{\xi}\ 2\ \{0,1\}^n:\sum_{j=1}^{n}\xi_j\quad\Gamma\right\} \tag{5.65}$$

Accordingly, given $\boldsymbol{\xi}\ 2\ \Xi$, the actual weight of each item j is $\underline{w}_j+\xi_j\tilde{w}_j$.

Once the uncertainty reveals, the decision maker has to decide, for each knapsack, the set of

items to be packed (among those dispatched to the knapsack in the first stage) while respecting the capacity constraint. This leads to the following definition of the wait-and-see feasible space:

$$Y(\boldsymbol{x},\boldsymbol{\xi}) = \left\{ \boldsymbol{y} \in \{0,1\}^{n \times K} : \begin{array}{ll} y_{jk} \leq x_{jk} & j = 1,...,n, k = 1,...,K \\ \sum_{j=1}^{n} (\underline{w}_j + \xi_j \tilde{w}_j) y_{jk} \leq W & k = 1,...,K \end{array} \right\} \tag{5.66}$$

5.6.1 Reformulation

By exploiting the presented reformulation, we have that the ARMKP is equivalent to the following problem with objective uncertainty only:

$$\min_{\boldsymbol{x} \in X} \max_{\boldsymbol{\xi} \in \Xi} \min_{(\boldsymbol{y},\boldsymbol{z}) \in Z^0(\boldsymbol{x})} \left\{ \sum_{j=1}^{n} \sum_{k=1}^{K} (-p_j y_{jk} + p_j \xi_j (y_{jk} - z_{jk})) \right\} \tag{5.67}$$

and

$$Z^0(\boldsymbol{x}) = \left\{ (\boldsymbol{y},\boldsymbol{z}) : \begin{array}{ll} \boldsymbol{y} \in \{0,1\}^{n \times K}, \boldsymbol{z} \in \{0,1\}^{n \times K} & j = 1,...,n, k = 1,...,K \\ z_{jk} \leq y_{jk} \leq x_{jk} & j = 1,...,n, k = 1,...,K \\ \sum_{j=1}^{n} (\underline{w}_j y_{jk} + \tilde{w}_j z_{jk}) \leq W & k = 1,...,K \end{array} \right\} \tag{5.68}$$

Given the here-and-now decisions, each item j can be packed in a knapsack k only if $x_{jk} = 1$. Intuitively, the objective function is well understood as it reduces to rewarding the selection of an item j by $-p_j$ if $\xi_j = 0$. If instead $\xi_j = 1$ (i.e., if the item should have a larger weight), then the decision maker can still achieve a reward of $-p_j$ provided that she decides $z_{jk} = 1$ as well (i.e., the item is taken with its additional weight). This is, indeed, the essence of Corollary 5.

5.6.2 Instance generation

We generated random ARMKP instances using the following input parameters: number of items $n \in \{10, 15\}$; uncertainty parameters $H \in \{0.1, 1.0\}$ and $\Gamma \in \{1, 2, 3, 4\}$; number of knapsacks $K \in \{2, 3, 4\}$; and capacity tightness ratio $\alpha \in \{0.25, 0.50, 0.75\}$.

For each item j, both the profit p_j and the nominal weight $\underline{w}_j$ were generated according to a discrete uniform distribution in $[1, 1000]$. The item weight in the worst case was defined as $\bar{w}_j = \underline{w}_j(1 + \delta_j)$, where δ_j was randomly generated in $[0, H]$, rounding the resulting value to the closest integer. In our instances, all knapsacks have the same capacity, defined as $\alpha \sum_{j=1}^{n} \underline{w}_j$. For each combination of these parameters, we generated 5 instances, thus producing a benchmark with 840 instances.

5.6.3 Results

Table 5.1 reports the outcome of our experiments. Column "opt" gives the number of instances (out of 5) that are solved to proven optimality within the time limit, whereas "time" reports the average computing time, computed with respect to the instances solved to optimality only.

			$K=2$				$K=3$				$K=4$			
			$H=0.1$		$H=1.0$		$H=0.1$		$H=1.0$		$H=0.1$		$H=1.0$	
n	α	Γ	opt	time	opt	time	opt	time	opt	time	opt	time	opt	time
10	0.25	1	5	0.4	5	0.5	5	6.0	5	2.1	5	405.6	5	70.1
		2	5	3.3	5	1.0	5	277.7	5	113.1	4	72.4	3	0.9
		3	5	6.4	5	5.3	5	209.7	5	86.3	4	1.1	4	0.8
		4	5	8.1	5	6.1	5	37.0	5	255.9	4	1.4	4	1.3
	0.50	1	5	3.7	5	0.7	5	126.0	5	241.3	5	0.2	5	1.0
		2	5	15.0	5	13.3	5	776.8	5	732.5	1	0.1	2	0.1
		3	5	56.8	5	16.7	3	886.3	5	1483.0	1	0.8	2	0.1
		4	5	69.1	5	56.6	5	1542.8	5	1400.1	1	1.5	2	0.4
	0.75	1	5	0.1	5	3.8	5	0.1	5	0.1	5	0.1	5	0.3
		2	5	37.6	5	22.4	5	0.8	5	749.4	5	0.6	5	5.8
		3	5	93.7	5	26.0	5	17.6	4	1357.0	5	2.6	5	22.8
		4	5	68.3	5	21.8	5	73.2	3	1198.8	5	9.7	5	87.2
15	0.25	1	5	4.3	5	5.6	1	1572.9	3	419.1	1	0.6	5	8.3
		2	5	253.5	5	132.6	0	–	1	3591.0	0	–	4	2.7
		3	5	658.0	5	588.4	0	–	0	–	0	–	4	40.2
		4	3	451.7	5	1041.6	2	2070.9	0	–	0	–	1	6.5
	0.50	1	5	173.7	5	32.7	5	0.8	3	68.6	5	0.2	5	0.4
		2	5	753.5	5	1008.1	0	–	0	–	5	208.3	1	686.1
		3	1	126.5	4	1379.1	0	–	0	–	0	–	0	–
		4	2	2369.9	3	1461.9	0	–	0	–	0	–	0	–
	0.75	1	5	0.1	5	0.1	5	0.1	5	0.1	5	0.1	5	0.1
		2	4	578.2	1	2314.7	5	2.1	5	72.6	5	1.2	5	11.0
		3	1	486.0	0	–	5	41.3	4	1178.5	5	61.8	5	105.5
		4	0	–	0	–	2	817.1	0	–	5	804.3	1	1235.2

Table 5.1: Computational results on ARMKP instances.

The results show that our algorithm is able to solve almost all the instances with $n = 10$, though some of them are far from trivial, in particular for $K = 4$ and $\alpha \in \{0.25, 0.75\}$. The problem appears to be easier for smaller values of Γ, in particular, for $\Gamma \in \{1, 2\}$. For $n = 15$ the instances are typically harder, in particular when $\Gamma = 3, 4$, i.e., approximately a quarter of the items can deviate from the nominal weight. For $K = 3$ and $\alpha \in \{0.25, 0.50\}$, only 15 out of 80 instances can be solved within time limit.

Table 5.2 gives some additional statistics on the behaviour of the algorithm, and reports the number of nodes which were generated during the execution of the branch-and-bound algorithm (column "nodes"), the average number of generated cuts (column "cuts"), and the average time spent for solving the relaxation and the separation problems (columns "t_{rel}" and "t_{sep}", respectively). All these figures are computed with respect to the instances solved to optimality only. These results show that, although we considered instances of medium size, the number of nodes is typically quite large (more than 1200, on average), each node producing, on average, around 60 separated cuts. In addition, we notice that the solving the relaxed problem is the most time consuming step of the algorithm, as it accounts for almost 90% of the total solution time.

5.7 Conclusion

In this chapter, we considered adjustable robust optimization problems with mixed-integer wait-and-see decisions and discrete uncertainty set. For this class of problems, we proposed a novel

n	α	Γ	opt	time	nodes	cuts	t_{rel}	t_{sep}
10	0.25	1	30	80.8	1597.0	3103.5	73.6	7.1
		2	27	84.0	1398.5	6957.5	78.3	5.7
		3	28	55.2	730.3	4960.5	52.6	2.6
		4	28	55.2	605.6	4639.6	52.3	2.9
	0.50	1	30	62.2	1292.6	5668.2	59.6	2.5
		2	23	334.3	2107.1	23262.9	314.1	20.2
		3	21	497.3	2207.8	33429.4	475.0	22.3
		4	23	667.2	2374.0	41392.0	626.4	40.7
	0.75	1	30	0.7	42.1	195.1	0.7	0.0
		2	30	136.1	635.9	10553.4	133.9	2.2
		3	29	215.2	528.6	16955.0	212.8	2.4
		4	28	174.9	264.1	12433.4	173.4	1.5
15	0.25	1	20	146.1	1426.3	6329.8	113.3	32.8
		2	15	368.8	2098.3	19669.2	293.5	75.3
		3	14	456.6	1612.9	33544.8	399.5	57.1
		4	11	973.8	2122.5	53311.2	911.0	62.8
	0.50	1	28	44.5	790.7	4955.1	42.1	2.3
		2	16	658.5	2300.9	45363.0	632.9	25.6
		3	5	1128.6	3365.8	71782.6	1120.4	8.2
		4	5	1825.1	2263.0	76708.6	1814.3	10.8
	0.75	1	30	0.1	1.3	20.3	0.1	0.0
		2	25	202.5	285.7	14262.9	200.3	2.1
		3	20	312.1	29.8	7396.3	310.9	1.2
		4	8	861.4	7.0	5828.5	858.4	3.0

Table 5.2: Additional statistics on ARMKP instances

reformulation in which uncertainty appears in the objective function only. This allows us to derive the first viable exact algorithm for this class of problems, closing a gap both from a theoretical and from a computational viewpoint with respect to the existing literature. We performed a computational analysis of the algorithm on an adjustable robust variant of the well-known combinatorial optimization problem coined the Adjustable Robust Multiple Knapsack Problem. Our computational results show that our approach is able to solve instances of medium size in a reasonable amount of time.

CHAPTER 6

Application: Facility Location Problem with uncertain demands

Henri Lefebvre, Enrico Malaguti, Michele Monaci[1]

Facility Location Problems (FLPs) are among the most prominent applications of operations research. This type of problem consists, for a company, in deciding the locations for opening facilities so as to serve a given set of clients with maximum efficiency. The vast literature dedicated to FLPs attest that a lot of variants can be considered given this definition.

In this chapter, we study FLPs in which the demand of each client is not completely known at decision time, as it typically happens in many practical applications. Moreover, the timing of the taken decisions (i.e., opening facilities and serving clients) suggests a two-stage nature of the decision flow. As a consequence, we consider the case where only the opening of facilities shall be decided here and now, the actual assignment of clients to opened facilities being postponed at a later instant. We then apply the theoretical results introduced in Chapter 5 in order to readily solve the obtained problem.

6.1 Problem description

6.1.1 Deterministic problem

We denote by V_1 be the set of candidate locations for opening facilities and V_2 be the set of clients. Each location $u \in V_1$ is associated with a setup cost f_u which must be paid for opening a facility in site u, as well as a maximum capacity q_u restricting the amount of goods leaving the site. Each client $v \in V_2$ is associated with a given demand, noted d_v and we let p_v denote the unitary profit earned by the decision maker for delivering one unit of product to a client. As it often happens in practical applications, partially serving a client is not allowed. For each connection $(u, v) \in V_1 \times V_2$, we denote by t_{uv} the unitary transportation cost from u to v.

[1] The content of this chapter has been accepted as a conference paper at *ROADEF 2022* which took place in Lyon, France.

We first assume that every input data is known at decision time. To model our problem, we introduce, for each site $u \in V_1$, a binary variable x_u which equals 1 if and only if a facility is opened in site u. Then, for every connection $(u,v) \in V_1 \times V_2$, we let $s_{uv} \geq 0$ be a continuous decision variable accounting for the amount of demand which travels from u to v. Finally, we let y_v be a binary variable, defined for each client $v \in V_2$, which equals 1 when client v is fully served by the company. Now, our Facility Location Problem can be modeled as follows.

$$\min \sum_{u \in V_1} f_u x_u + \sum_{u \in V_1} \sum_{v \in V_2} t_{uv} s_{uv} - \sum_{v \in V_2} p_v d_v y_v \tag{6.1}$$

$$\text{s.t.} \sum_{u \in V_1} s_{uv} = d_v y_v \qquad \forall v \in V_2 \tag{6.2}$$

$$\sum_{v \in V_2} s_{uv} \leq q_u x_u \qquad \forall u \in V_1 \tag{6.3}$$

$$s_{uv} \geq 0 \qquad \forall (u,v) \in V_1 \times V_2 \tag{6.4}$$

$$x_u \in \{0,1\} \qquad \forall u \in V_1 \tag{6.5}$$

$$y_v \in \{0,1\} \qquad \forall v \in V_2 \tag{6.6}$$

Here, constraints (6.2) enforces that clients which are chosen to be served have their demand fulfilled, while constraints (6.3) ensures that the total capacity of each facility is not exceeded. Finally, the objective function (6.1) minimizes the sum of the opening costs and transportation costs, to which is removed the profit made fulfilling each clients' demand. Note that we trivially have, for each $(u,v) \in V_1 \times V_2$, $s_{uv} \leq q_u$ so that every involved variables are generically bounded. Moreover, observe that (6.2) can be turned into a greater-or-equal-to constraint without changing the optimal solution space while dividing by two the size of the dual space of the continuous relaxation of our model.

6.1.2 A two-stage robust variant

In practical applications, assuming knowing the exact demand of each client when taking the decision to open facilities is a demanding assumption. To circumvent this fact, we introduce a two-stage robust variant of FLP where demands are uncertain. In particular, following the guidelines of Bertsimas and Sim [2004], we assume that the demand of each client v has a nominal value $\bar{d}_v$ and can be either increased by a fixed amount $\tilde{d}_v^+$ or decreased by a fixed amount $\tilde{d}_v^-$. For the sake of simplicity, we consider the case in which $\tilde{d}_v^+ = \tilde{d}_v^-$ and denote by $\tilde{d}_v$ this value. We assume that up to Γ clients can change their demands, where Γ is an integer input parameter used to control the robustness of the solution. Accordingly, each scenario is characterized by two subsets $L \subseteq V_2$ and $H \subseteq V_2$ with $|L| + |H| \leq \Gamma$ and the actual demand of each client $v \in L$ (resp. $v \in H$) is $\bar{d}_v - \tilde{d}_v$ (resp. $\bar{d}_v + \tilde{d}_v$), all remaining clients (i.e., in $V_2 \setminus (L \cup H)$) having a demand equal to $\bar{d}_v$. We introduce set Ξ, modeling the uncertainty set as

$$\Xi = \{(L,H) : L \subseteq V_2, H \subseteq V_2, L \cap H = \emptyset, |L| + |H| \leq \Gamma\} \tag{6.7}$$

As done in Section 6.1.1 for the deterministic case, we model the here-and-now decisions by introducing $|V_1|$ binary decision variables such that, for $u \in V_1$, $x_u = 1$ iff a facility is opened in

site u. We note $X = \{0,1\}^{|V_1|}$.

The wait-and-see (assignment) problem is modeled as follows: for each client $v \in V_2$, we introduce a binary variable y_v taking value 1 iff client v is entirely served and, for each connection $(u,v) \in V_1 \times V_2$, there is a non-negative continuous variable s_{uv} representing the amount of goods transported from u to v. Each client v is entirely served if the amount of goods arriving to v is equal to its actual demand, see constraints (6.8). Moreover, constraints (6.9) impose that the total amount of goods leaving each site $u \in V_1$ must not exceed its total capacity q_u. Therefore, for a fixed here-and-now decision $\boldsymbol{x} \in X$ and for given L and H (as described above), the feasible space $Y(\boldsymbol{x}, L, H)$ for the wait-and-see variables includes all vectors $(\boldsymbol{y}, \boldsymbol{s}) \in \{0,1\}^{|V_2|} \times \mathbb{R}_+^{|V_1| \times |V_2|}$ fulfilling the following constraints:

$$\sum_{u \in V_1} s_{uv} \quad \bar{d}_v y_v + \begin{cases} \tilde{d}_v y_v & \text{if } v \in L \\ +\tilde{d}_v y_v & \text{if } v \in H \end{cases} \qquad \forall v \in V_2 \tag{6.8}$$

$$\sum_{v \in V_2} s_{uv} \quad q_u x_u \qquad \forall u \in V_1 \tag{6.9}$$

Finally, the two-stage robust facility location problem which we consider is the following

$$\min_{\boldsymbol{x} \in X} \left\{ \sum_{u \in V_1} f_u x_u + \max_{(L,H) \in \Xi} \min_{(\boldsymbol{y},\boldsymbol{s}) \in Y(\boldsymbol{x},L,H)} F(\boldsymbol{y}, \boldsymbol{s}, L, H) \right\}, \tag{6.10}$$

where function F is defined as

$$F(\boldsymbol{y}, \boldsymbol{s}, L, H) = \sum_{(u,v) \in V_1 \times V_2} t_{uv} s_{uv} \quad \sum_{v \in L} p_v (d_v - \bar{d}_v) y_v \quad \sum_{v \in V_2 \setminus (L \cup H)} p_v d_v y_v \quad \sum_{v \in H} p_v (d_v + \tilde{d}_v) y_v \tag{6.11}$$

Before, jumping to the reformulation of problem (6.10) as a two-stage robust problem with objective uncertainty (applying the results introduced in 5), we discuss, in the next section, a numerical example showing the relevance of considering uncertain demands in this context.

6.2 Numerical example

Let us now consider the numerical example presented in Figure 6.1a where the numbers close to the arcs denote the unitary transportation costs between sites (rectangles) and clients (circles). The figure also reports the capacity and opening cost of each facility, as well as the unitary profit and demand for each client.

Assuming that no client changes his demand, the optimal solution is to open two facilities in sites 1 and 3 in order to both serve client A and B. The associated profit is 623, as it can be seen in Figure 6.1b where figures on the arcs are the amount of goods being transported (opening costs: $1286 + 867$, transportation costs: $106 \quad 7 + 20 \quad 6 + 74 \quad 3$, earnings: $106 \quad 24 + 94 \quad 14$; profit: 623). Yet, we show that this solution is not robust when uncertainty is at stake. Indeed, if client B increases his demand to 116, the total demand ($106 + 116 = 222$) becomes greater than the overall capacity of the opened facilities ($143 + 74 = 217$). For this reason, one of the two clients cannot be served anymore. Even worse, the decision maker won't be able to make profit in this particular setting. Indeed, assume that the decision maker decides to serve client A

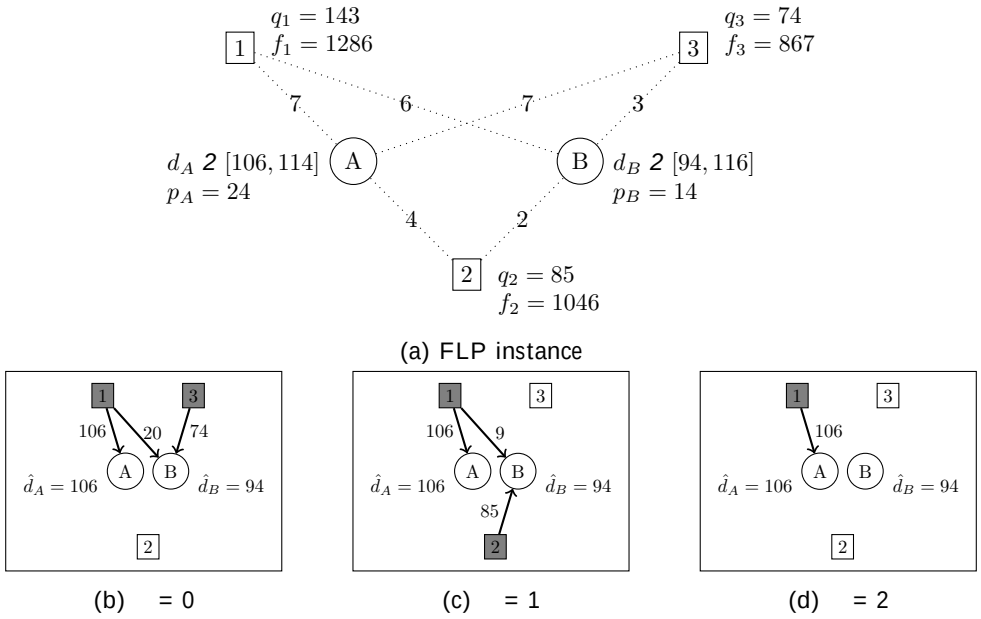


Figure 6.1: Robust solutions for FLP with different uncertainty budgets Γ

only. This choice would yield a negative profit of 351 (already-paid opening costs: 1286 + 1046, transportation costs: 106 7, earnings: 106 24; profit: 351). On the other hand, serving client B only leads to a negative profit of 1143 (already-paid opening costs: 1286+1046, transportation costs: 74 3 + 42 6, earnings: 116 14; profit: 1143). All in all, we end up in a situation which is not profitable as a useless facility has been opened.

On the other hand, the decision maker could take into account the uncertain demands when designing its network. Assuming that at most one client changes his demand, a more robust solution is to open facilities in site 1 and 2, so as to prevent the previous situation to occur. In such design, the worst-case scenario is one in which no client change his demand and the associated profit is then of 562, achieved by serving both clients (opening costs: 1286 + 1046, transportation costs: 106 7 + 31 6 + 85 2, earnings: 106 24 + 94 14; profit: 562). This situation is depicted in Figure 6.1c. Now, assume that client B increases his demand to 116, as we previously imagined. In this case, serving client B remains feasible, and the associated profit is 738 (opening costs: 1286 + 1046, transportation costs: 106 7 + 31 6 + 85 2, earnings: 106 24 + 116 14; profit: 738).

In Figure 6.1d, we depicted the optimal here-and-now decision as well as the optimal wait-and-see decision in the worst case assuming that up to two clients change their demands. In this case, the worst-case corresponds to having both clients with their nominal demands. Note that the here-and-now decision, again, changes to lead more robust decisions. Under this assumption, in the worst-case, the decision maker will make a profit of 516.

This small example shows the importance and practical relevance of considering robust approaches as sensitivity analysis for network designs in FLP contexts.

6.3 Reformulation

In this section, we show how to apply the results obtained in Chapter 6 to our two-stage robust FLP variant. Results are stated without proof as they directly follow from the previous chapter. Let us start by considering the binary encoded version of Ξ. We consider the binary set $\Xi \subseteq \{0,1\}^{|V_2|} \times \{0,1\}^{|V_2|}$ comprising all couples of vectors $\boldsymbol{l}$ and $\boldsymbol{h}$ such that the following constraints are satisfied.

$$\sum_{v \in V_2} (l_v + h_v) \leq \Gamma \text{ and } l_v + h_v \leq 1 \quad \forall v \in V_2 \tag{6.12}$$

Informally, $(\boldsymbol{l}, \boldsymbol{h})$ now encodes sets L and H as binary decision variables, and one can consider $Y(\boldsymbol{x}, \boldsymbol{l}, \boldsymbol{h})$ equivalent to $Y(\boldsymbol{x}, L, H)$. This allows us to rewrite the inner minimization problem (i.e., the wait-and-see problem) as follows, in terms of $\boldsymbol{x}, \boldsymbol{l}$ and $\boldsymbol{h}$.

$$\min \sum_{v \in V_2} \left(\sum_{u \in V_1} t_{uv} s_{uv} \leq p_v(\bar{d}_v + \tilde{d}_v l_v + \tilde{d}_v h_v) y_v \right) \tag{6.13}$$

$$\text{s.t.} \sum_{u \in V_1} s_{uv} \leq y_v(\bar{d}_v + \tilde{d}_v l_v + \tilde{d}_v h_v) \qquad \forall v \in V_2 \tag{6.14}$$

$$(6.9)$$

$$\boldsymbol{s} \in \mathbb{R}_+^{|V_1| \times |V_2|}, \boldsymbol{y} \in \{0,1\}^{|V_2|}$$

In this problem, uncertainty appears in the objective function and each uncertain coefficient may take three values. As discussed in Section 5.2.2, this case can be incorporated in the framework presented in Chapter 5. However, in this specific case where each coefficient can take 3 values only, a more direct reformulation can be obtained by introducing variables $\boldsymbol{z}^l$ and $\boldsymbol{z}^h$ such that $z_v^l = y_v l_v$ and $z_v^h = y_v h_v$ (for all $v \in V_2$) and adding the corresponding linearization constraints

$$z_v^l \leq l_v \quad z_v^l \leq y_v \quad z_v^l \geq y_v + l_v - 1 \quad z_v^h \leq h_v \quad z_v^h \leq y_v \quad z_v^h \geq y_v + h_v - 1 \tag{6.15}$$

Let us introduce, for all $\boldsymbol{x} \in X$ and all $(\boldsymbol{h}, \boldsymbol{l}) \in \Xi$, set $Z(\boldsymbol{x}, \boldsymbol{l}, \boldsymbol{h})$ as the set of decision variables $(\boldsymbol{y}, \boldsymbol{S}, \boldsymbol{z}^l, \boldsymbol{z}^h)$ such that $\boldsymbol{y} \in \{0,1\}^{|V_2|}$, $\boldsymbol{S} \in \mathbb{R}_+^{|V_1| \times |V_2|}$, $\boldsymbol{z}^l \in \{0,1\}_+^{|V_2|}$ and $\boldsymbol{z}^h \in \{0,1\}_+^{|V_2|}$ and fulfilling capacity constraints (6.9), linearization constraints (6.15) as well as constraints (6.16) defined as (6.14) where bilinear terms have been substituted.

$$\sum_{u \in V_1} s_{uv} \leq \bar{d}_v y_v + \tilde{d}_v z_v^l + \tilde{d}_v z_v^h \quad v \in V_2 \tag{6.16}$$

In turn, it is clear that the wait-and-see problem is equivalent to the following problem.

$$\min_{(\boldsymbol{y}, \boldsymbol{S}, \boldsymbol{z}^l, \boldsymbol{z}^h) \in Z(\boldsymbol{x}, \boldsymbol{l}, \boldsymbol{h})} \left\{ \sum_{v \in V_2} \left(\sum_{u \in V_1} t_{uv} s_{uv} \leq p_v(\bar{d}_v + \tilde{d}_v l_v + \tilde{d}_v h_v) y_v \right) \right\} \tag{6.17}$$

By the same arguments used in Remark 14, constraints of type "$z_v^h \leq h_v$" and of type "$z_v^l \geq y_v + l_v - 1$" can be omitted.

Now, following the development of Chapter 5, we can derive the following theorem.

Theorem 7. *For all $x \in X$ and $(l, h) \in \Xi$, the wait-and-see problem in the two-stage robust FLP is equivalent the following problem,*

$$\max_{(\lambda^l, \lambda^h) \geq 0} g(\lambda^l, \lambda^h, x, l, h) \tag{6.18}$$

where $g(\lambda^l, \lambda^h; x, l, h)$ is defined as the optimal objective value of the following problem,

$$\min_{(y, S, z^l, z^h) \in Z_X(x)} \left\{ \sum_{v \in V_2} \left(\sum_{u \in V_1} t_{uv} s_{uv} - p_v(d_v - \hat{d}_v l_v + \hat{d}_v h_v) y_v + \lambda_v^h h_v (y_v - z_v^h) + (1 - l_v) \lambda_v^l z_v^l \right) \right\} \tag{6.19}$$

where $Z_X()$ is defined as $Z()$ where constraints "$z_v^h \leq h_v$", "$z_v^l \leq y_v + l_v \leq 1$", "$y_v \leq z_v^l \leq 1 - l_v$" and "$z_v^h \leq y_v + h_v \leq 1$" have been omitted.

Moreover, a clever analysis of the obtained problem in Theorem 7 directly leads to the following corollary, akin to Corollary 5 and Lemma 5.

Corrolary 6. *For all $v \in V_2$, let $\underline{\lambda}_j^l = p_j(\bar{d}_j - \tilde{d}_j)$ and $\underline{\lambda}_j^h = p_j(\bar{d}_j + \tilde{d}_j)$, then, the following holds.*

$$\max_{(\lambda^l, \lambda^h) \geq 0} g(\lambda^l, \lambda^h, x, l, h) = g(\underline{\lambda}^l, \underline{\lambda}^h; x, l, h) \tag{6.20}$$

All in all, we have now obtained the desired reformulation of our two-stage robust FLP into an equivalent two-stage robust problem where the uncertain parameters are confined in the objective function. This reformulation is recalled as follows.

$$\min_{x \in X} \max_{(l,h) \in \Xi} \min_{(y, S, z^l, z^h) \in Z_X(x)} \Pi(y, S, z^l, z^h, l, h) \tag{6.21}$$

$$\Pi(y, S, z^l, z^h, l, h) = \sum_{j \in V_2} \left(\sum_{u \in V_1} t_{uv} s_{uv} - p_v(d_v - \hat{d}_v l_v + \hat{d}_v h_v) y_v + \underline{\lambda}_v^h h_v (y_v - z_v^h) + (1 - l_v) \underline{\lambda}_v^l z_v^l \right) \tag{6.22}$$

Then, the algorithmic solution from Kämmerling and Kurtz [2020], which was extended in Chapter 5, can be employed in order to solve the obtained reformulation.

6.4 Computational experiments

6.4.1 Instance generation

We generated facility location instances according to Cornuéjols et al. [1991], and considered the following sizes ($|V_1|, |V_2|$): (6, 12), (8, 16), (10, 12), and (12, 24). For each site $u \in V_1$, the capacity q_u was uniformly generated between 10 and 160 while the opening cost was computed as $f_u = \alpha_u + \beta_u \sqrt{q_u}$ where α_u and β_u were generated between 0 and 90 and 100 and 110, respectively. The candidate positions for opening facilities and the location of clients were randomly generated in the unitary square. Then, for each pair $(u, v) \in V_1 \times V_2$, the transportation cost t_{uv} was defined as the associated Euclidean distance multiplied by 10. Demands were generated so that $\sum_{u \in V_1} q_u / \sum_{v \in V_2} \bar{d}_v = \mu$ where μ is a parameter taking value 1.5 or 2, and $\tilde{d}_v$ was set to $\bar{d}_v$ multiplied by a randomly generated number between 0.00 and 0.25 (i.e., demands can vary up to 25%). Finally, every client's profit was set equal to $p_v = \frac{4}{|V_1|} \sum_{u \in V_1} t_{uv}$. Every input data

			$\mu = 1.5$		$\mu = 2.0$	
$\|V_1\|$	$\|V_2\|$	Γ	opt	time	opt	time
6	12	2	16	0.9	16	0.8
		4	16	20.6	16	29.5
		6	16	117.9	15	107.0
8	16	2	16	3.5	16	2.8
		4	15	367.4	15	173.9
		6	5	143.7	11	845.5
10	20	2	16	9.4	16	6.4
		4	11	752.1	14	549.3
		6	3	1150.2	7	1123.1
12	24	2	16	18.6	16	15.7
		4	9	1277.1	5	797.1
		6	2	708.7	1	2173.8

Table 6.1: Computational results on ARFLP instances

was rounded to the closest integer. For each combination of $(|V_1|, |V_2|)$ and μ, we generated 16 instances which were solved for $\Gamma \in \{2, 6, 8\}$.

6.4.2 Results

Tables 6.1 and 6.2 report the outcome of our experiments and give the same information as Tables 5.1 and 5.2 in Chapter 5. The results show that our reformulation is able to solve most of the instances of size $(6, 12)$ and $(8, 16)$ for relatively small values of Γ (2 and 4 in particular), whereas larger instances seem to be harder to solve within the given time limit. In addition, we note that our approach can solve all instances with $\Gamma = 2$ in less than twenty seconds. Finally, we can notice that the instances with bigger values of μ are harder to solve in practice. As to the number of nodes, it is considerably smaller than its counterpart on the ARMKP instances, the average value being 2. On average, the algorithm generates 600 cuts per node, i.e., considerably more than in the ARMKP case. Finally, for this problem as well, the solution of the relaxed problem takes most of the total time (around 80% on average), the separation problem requiring a smaller computational effort. Indeed, while the relaxation consists in maximizing a concave (piecewise) function over a binary set, the separation problem has the same complexity as the deterministic FLP, which can be easily solved in practice for medium-size instances.

$\|V_1\|$	$\|V_2\|$	Γ	opt	time	nodes	cuts	t_{rel}	t_{sep}
6	12	2	32	0.9	2.3	72.8	0.4	0.4
		4	32	25.1	2.3	400.8	21.5	3.4
		6	31	112.6	2.5	732.4	104.1	8.2
8	16	2	32	3.2	2.3	146.1	1.9	1.2
		4	30	270.6	2.3	1143.1	249.3	20.6
		6	16	626.1	1.6	1378.6	591.5	33.8
10	20	2	32	7.9	2.6	230.8	5.1	2.6
		4	25	638.5	2.4	1595.2	587.6	49.6
		6	10	1131.3	1.0	1466.4	1069.4	60.7
12	24	2	32	17.1	2.1	312.4	11.6	5.2
		4	14	1105.7	1.7	1931.4	1012.8	90.7
		6	3	1197.1	1.0	1648.0	1113.2	82.2

Table 6.2: Additional statistics on ARFLP instances

CHAPTER 7

Convex problems with 0-1 polytope uncertainty

Henri Lefebvre, Enrico Malaguti, Michele Monaci

In this chapter, we study ARO problems where the second-stage feasible space is defined by means of convex functions. Such ARO problems can be formulated as

$$\inf_{\boldsymbol{x} \in X} \sup_{\boldsymbol{\xi} \in \Xi} \inf_{\boldsymbol{y} \in Y(\boldsymbol{x},\boldsymbol{\xi})} g_0(\boldsymbol{x}, \boldsymbol{y}), \tag{7.1}$$

where X denotes the feasible set of decisions to be taken here-and-now (first-stage decisions), Ξ is the uncertainty set, and $Y(\boldsymbol{x}, \boldsymbol{\xi})$ is the set of all feasible recourse actions (second-stage decisions) for a given $\boldsymbol{x} \in X$ and $\boldsymbol{\xi} \in \Xi$.

7.1 Introduction

7.1.1 Problem formulation

Let n_X, n_Ξ, n_Y and m be given natural numbers. We let the first-stage feasible set X be any subset of $\mathbb{R}^{n_X}$ and assume that the uncertainty set $\Xi \subseteq \mathbb{R}^{n_\Xi}$ is defined as

$$\Xi = \left\{ \boldsymbol{\xi} \in \mathbb{R}^{n_\Xi}_+ : \boldsymbol{U}\boldsymbol{\xi} \geq \boldsymbol{d} \right\} \tag{7.2}$$

where $\boldsymbol{U} \in \mathbb{R}^{q \times n_\Xi}$ and $\boldsymbol{d} \in \mathbb{R}^{n_q}$. As is customary, we write matrices and vectors in bold case and use $\geq$ to compare vectors of agreeable size.

For each $i = 1, \ldots, m$ and $j = 1, \ldots, n_\Xi$, we let $f_{ij} : \mathbb{R}^{n_X} \to \mathbb{R}$ be given real-valued convex functions; for a given $\boldsymbol{x} \in \mathbb{R}^{n_X}$, we denote by $\boldsymbol{F}(\boldsymbol{x})$ the $m \times n_\Xi$ matrix whose generic element is $f_{ij}(\boldsymbol{x})$. Similarly, for each $i = 1, \ldots, m$, we let $g_i : \mathbb{R}^{n_X + n_Y} \to \mathbb{R}$ be a convex function; for a given $\boldsymbol{x} \in \mathbb{R}^{n_X}$ and $\boldsymbol{y} \in \mathbb{R}^{n_Y}$, we denote by $\boldsymbol{g}(\boldsymbol{x}, \boldsymbol{y})$ the m-dimensional vector whose generic element is $g_i(\boldsymbol{x}, \boldsymbol{y})$.

For given $\boldsymbol{x} \in X$ and $\boldsymbol{\xi} \in \Xi$, the second-stage feasible space $Y(\boldsymbol{x}, \boldsymbol{\xi})$ is defined by

$$Y(\boldsymbol{x}, \boldsymbol{\xi}) = \{\boldsymbol{y} \in \mathbb{R}^{n_Y} : \boldsymbol{F}(\boldsymbol{x})\boldsymbol{\xi} + \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{y}) \geq \mathbf{0}\}. \tag{7.3}$$

Finally, we assume that the objective function $g_0 : \mathbb{R}^{n_X+n_Y} \to \mathbb{R}$ is a real-valued convex function. In other words, we consider the class of problems of type (7.1) where the uncertainty set has a polyhedral representation and the second-stage optimization problem is modelled as a convex problem.

Additionally, we make the following technical assumption.

Assumption O. *For every $(x_0, \boldsymbol{x}) \in \mathbb{R} \times X$ and every $\boldsymbol{\xi} \in \Xi$, the following set, noted $Z(\boldsymbol{x}, \boldsymbol{\xi})$, is closed.*

$$Z(x_0, \boldsymbol{x}, \boldsymbol{\xi}) = \left\{ \begin{pmatrix} \alpha \\ \boldsymbol{\beta} \end{pmatrix} \in \mathbb{R}^{m+1} : \exists \boldsymbol{y} \in \mathbb{R}^{n_Y}, \begin{array}{ll} g_0(\boldsymbol{x}, \boldsymbol{y}) & x_0 \quad \alpha \\ \boldsymbol{F}(\boldsymbol{x})\boldsymbol{\xi} + \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{y}) & \boldsymbol{\beta} \end{array} \right\} \tag{7.4}$$

7.1.2 Contribution

As discussed in the literature review in Chapter 2, few exact methods for convex ARO have been proposed so far, mostly relying on strong assumptions. In this work, we start filling this gap by providing a reformulation which is valid for the broad case of ARO problems introduced above. In addition, we introduce a Generalized-Benders-Decomposition-type algorithm Geoffrion [1972] and a column-and-constraint generation scheme Zhao and Zeng [2012] under the unifying lens of convex conjugates. This method can be applied to any convex ARO, including cases where the second stage is an SOCP, an SDP, or a (conic) LP. In our solution scheme, separation of robust feasible solutions asks for solving a non-convex optimization problem. We show that a suitable convex reformulation of the separation problem can be used when the uncertainty set has some structural properties, thus allowing to derive a computationally sound algorithm built on top of a general-purpose solver. Finally, we give computational evidence of the applicability of our solution method to an uncertain planing application from the literature.

This chapter is organized as follows. In Section 7.2, we introduce our main theoretical results. In particular, we show in Section 7.2.1 that separating first-stage solutions can be done by means of a non-convex MINLP and derive a Generalized Benders Decomposition algorithm in Section 7.2.2 and a column-and-constraint generation algorithm in Section 7.2.3. Convergence of both algorithms is established in Section 7.2.4. In Section 7.2.5, we show how the non-convex MINLP can be exactly reformulated as a convex MINLP by exploiting structural properties of the uncertainty set. Finally, we apply our solution methods to a resource planning problem in Section 7.3.

7.2 Theoretical development

7.2.1 A non-convex separation problem

Problem 7.1 can be reformulated as (see e.g., Takeda et al. [2007])

$$(P) \qquad \inf\ x_0 \tag{7.5}$$

$$\text{s.t. } \boldsymbol{x} \in X, x_0 \in \mathbb{R} \tag{7.6}$$

$$\forall \boldsymbol{\xi} \in \Xi, \exists \boldsymbol{y} \in Y(\boldsymbol{x}, \boldsymbol{\xi}), x_0 \quad g_0(\boldsymbol{x}, \boldsymbol{y}). \tag{7.7}$$

Since explicitly adding all constraints (7.7) to the formulation is not viable in practice, we follow a separation approach in which, given pair $(x_0, \boldsymbol{x})$, we check whether a violated constraint exists. Solving the *separation problem* asks to answer the following question:

"Given $(x_0, \boldsymbol{x}) \in \mathbb{R} \quad X$, can we show that for all $\boldsymbol{\xi} \in \Xi$ there exists a $\hat{\boldsymbol{y}} \in Y(\boldsymbol{x}, \boldsymbol{\xi})$ such that $x_0 \quad g_0(\boldsymbol{x}, \hat{\boldsymbol{y}})$? If not, can we identify $\hat{\boldsymbol{\xi}} \in \Xi$ such that either $Y(\boldsymbol{x}, \hat{\boldsymbol{\xi}}) = \emptyset$ or $\forall \boldsymbol{y} \in Y(\boldsymbol{x}, \hat{\boldsymbol{\xi}}), x_0 < g_0(\boldsymbol{x}, \boldsymbol{y})$?".

In the following lemma, we give a sufficient and necessary condition for answering an easier question: given $(x_0, \boldsymbol{x}) \in \mathbb{R} \quad X$ and $\boldsymbol{\xi} \in \Xi$, is there a feasible second-stage decision $\hat{\boldsymbol{y}} \in Y(\boldsymbol{x}, \boldsymbol{\xi})$ such that $x_0 \quad g_0(\boldsymbol{x}, \hat{\boldsymbol{y}})$?

Lemma 6. *Given $(x_0, \boldsymbol{x}) \in \mathbb{R} \quad X$ and $\boldsymbol{\xi} \in \Xi$, there exists $\hat{\boldsymbol{y}} \in Y(\boldsymbol{x}, \boldsymbol{\xi})$ such that $x_0 \quad g_0(\boldsymbol{x}, \hat{\boldsymbol{y}})$ if and only if the following condition holds*

$$\forall (\lambda_0, \boldsymbol{\lambda}) \in \mathbb{R}^{m_Y+1}_+, \quad \inf_{\boldsymbol{y} \in \mathbb{R}^{n_Y}} \left\{ \boldsymbol{\lambda}^T (\boldsymbol{F}(\boldsymbol{x})\boldsymbol{\xi} + \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{y})) + \lambda_0 (g_0(\boldsymbol{x}, \boldsymbol{y}) \quad x_0) \right\} \quad 0. \tag{7.8}$$

Proof. First, it is trivial that $\hat{\boldsymbol{y}} \in Y(\boldsymbol{x}, \boldsymbol{\xi})$ and $x_0 \quad g_0(\boldsymbol{x}, \hat{\boldsymbol{y}})$ implies (7.8).
Assume now that condition (7.8) holds, in which case we have

$$\sup_{(\lambda_0, \boldsymbol{\lambda}) \in \mathbb{R}^{m_Y+1}_+} \inf_{\boldsymbol{y} \in \mathbb{R}^{n_Y}} \left\{ \boldsymbol{\lambda}^T (\boldsymbol{F}(\boldsymbol{x})\boldsymbol{\xi} + \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{y})) + \lambda_0 (g_0(\boldsymbol{x}, \boldsymbol{y}) \quad x_0) \right\} \quad 0 \tag{7.9}$$

Since $(\mathbf{0}, 0)$ is a possible choice for $(\lambda_0, \boldsymbol{\lambda})$ in (7.9), we have that

$$\sup_{(\lambda_0, \boldsymbol{\lambda}) \quad \mathbf{0}} \inf_{\boldsymbol{y} \in \mathbb{R}^{n_Y}} \left\{ \boldsymbol{\lambda}^T (\boldsymbol{F}(\boldsymbol{x})\boldsymbol{\xi} + \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{y})) + \lambda_0 (g_0(\boldsymbol{x}, \boldsymbol{y}) \quad x_0) \right\} = 0. \tag{7.10}$$

This shows that the dual of the following problem has an optimal objective value of 0.

$$\inf_{\boldsymbol{y} \in \mathbb{R}^{n_Y}} \{0 : x_0 \quad g_0(\boldsymbol{x}, \boldsymbol{y}), \boldsymbol{y} \in Y(\boldsymbol{x}, \boldsymbol{\xi})\}. \tag{7.11}$$

Recalling Assumption O, it must be that the primal problem (7.11) is feasible (see also, Geoffrion [1972], Theorem 5.1).

□

Remark 18. *Condition* (7.8) *of Lemma 6 remains valid when adding the restriction $\|(\lambda_0, \boldsymbol{\lambda})\| = 1$, where $\|\ \|$ is any norm of $\mathbb{R}^{m+1}$. Indeed, scaling does not impact the sign of the inner inf optimization problem in* (7.9).

Thanks to Lemma 6, we now introduce a non-convex optimization problem which solves the separation problem.

Theorem 8. *Let $(x_0, \boldsymbol{x}) \in \mathbb{R} \quad X$, the following propositions are equivalent.*

1. $\forall \boldsymbol{\xi} \in \Xi, \exists \hat{\boldsymbol{y}} \in Y(\boldsymbol{x}, \boldsymbol{\xi}), x_0 \quad g_0(\boldsymbol{x}, \hat{\boldsymbol{y}})$;

2. *The following non-convex optimization problem has an optimal objective value which is non-positive*

$$\sup \quad \sum_{i=0}^{m} \lambda_i g_i \boldsymbol{j}_{\boldsymbol{x}} \left(\frac{\boldsymbol{u}^i}{\lambda_i} \right) + \boldsymbol{\lambda}^T \boldsymbol{F}(\boldsymbol{x}) \boldsymbol{\xi} \quad \lambda_0 x_0 \tag{7.12}$$

$$\text{s.t.} \sum_{i=0}^{m} \boldsymbol{u}^i = 0 \tag{7.13}$$

$$(\lambda_0, \boldsymbol{\lambda}) \; 2 \; \Lambda \tag{7.14}$$

$$\boldsymbol{\xi} \; 2 \; \Xi \tag{7.15}$$

$$\boldsymbol{u}^i \; 2 \; \mathsf{R}^{n_Y} \qquad i = 0, 1, \ldots, m, \tag{7.16}$$

where $g_i \boldsymbol{j}_{\boldsymbol{x}}(\;) = g_i(\boldsymbol{x}, \;)$, $g_0 \boldsymbol{j}_{\boldsymbol{x}}(\;) = g_0(\boldsymbol{x}, \;)$, *and* $\Lambda = f(\lambda_0, \boldsymbol{\lambda}) \; 2 \; \mathsf{R}^m_+ \quad \mathsf{R}_+ : jj(\lambda_0, \boldsymbol{\lambda}) jj = 1 g.$

Proof. Let $(x_0, \boldsymbol{x}) \; 2 \; \mathsf{R} \quad X$. By Lemma 6, for any $\boldsymbol{\xi} \; 2 \; \Xi$, there exists $\hat{\boldsymbol{y}} \; 2 \; Y(\boldsymbol{x}, \boldsymbol{\xi})$ such that $x_0 \quad g_0(\boldsymbol{x}, \hat{\boldsymbol{y}})$ if and only if condition (7.8) is satisfied. Let $(\lambda_0, \boldsymbol{\lambda})$ be any element of Λ. We start by re-arranging the terms of (7.8) for $(\lambda_0, \boldsymbol{\lambda})$ as follows

$$\inf_{\boldsymbol{y} 2 \mathsf{R}^{n_Y}} \left\{ \boldsymbol{\lambda}^T \boldsymbol{g} \boldsymbol{j}_{\boldsymbol{x}}(\boldsymbol{y}) + \lambda_0 g_0 \boldsymbol{j}_{\boldsymbol{x}}(\boldsymbol{y}) \right\} + \boldsymbol{\lambda}^T \boldsymbol{F}(\boldsymbol{x}) \boldsymbol{\xi} \quad \lambda_0 x_0 \quad 0 \tag{7.17}$$

where terms which do not depend on $\boldsymbol{y}$ are moved out from the optimization problem. Now, letting $\phi(\boldsymbol{y}) = \boldsymbol{\lambda}^T \boldsymbol{g} \boldsymbol{j}_{\boldsymbol{x}}(\boldsymbol{y}) + \lambda_0 g_0 \boldsymbol{j}_{\boldsymbol{x}}(\boldsymbol{y})$, by definition the inf problem in (7.17) is $(\quad \phi) \; (\boldsymbol{0})$. By exploiting the fact that $(\quad \phi) \; (\boldsymbol{z}) = \quad \phi \; (\quad \boldsymbol{z})$ for any $\boldsymbol{z}$ (see Rockafellar [1996], p. 308), we have that (7.17) is equivalent to

$$\phi \; (\boldsymbol{0}) + \boldsymbol{\lambda}^T \boldsymbol{F}(\boldsymbol{x}) \boldsymbol{\xi} \quad \lambda_0 x_0 \quad 0. \tag{7.18}$$

Using standard conjugate rules (see Rockafellar [1996], p. 145), one obtains the following expression of ϕ

$$\phi \; (\boldsymbol{z}) = \inf \; \sum_{i=0}^{m} (\lambda_i g_i \boldsymbol{j}_{\boldsymbol{x}}) \; (\boldsymbol{u}^i) \tag{7.19}$$

$$\text{s.t.} \sum_{i=0}^{m} \boldsymbol{u}^i = \boldsymbol{z} \tag{7.20}$$

$$\boldsymbol{u}^i \; 2 \; \mathsf{R}^{n_Y} \qquad i = 0, 1, ..., m. \tag{7.21}$$

Then, we have $(\lambda_i g_i \boldsymbol{j}_{\boldsymbol{x}}) \; (\boldsymbol{u}^i) = \lambda_i g_i \boldsymbol{j}_{\boldsymbol{x}}(\boldsymbol{u}^i / \lambda_i)$ (see Rockafellar [1996], p. 140). The proof is achieved by requiring that (7.17) be enforced for all $\boldsymbol{\xi} \; 2 \; \Xi$. □

Example 11 (ℓ_p-norm objective and constraints)**.** *Assume that, for each* $i = 0, 1, ..., m$, *it holds* $g_i(\boldsymbol{x}, \boldsymbol{y}) = \left|\left| \boldsymbol{K}_X^i \boldsymbol{x} + \boldsymbol{K}_Y^i \boldsymbol{y} + \boldsymbol{\chi}^i \right|\right|_{p_i} + \boldsymbol{\delta}_X^{i\,T} \boldsymbol{x} + \boldsymbol{\delta}_Y^{i\,T} \boldsymbol{y} + \kappa^i$ *where* $\boldsymbol{K}_X^i$, $\boldsymbol{K}_Y^i$, $\boldsymbol{\chi}^i$, $\boldsymbol{\delta}_X^i$, $\boldsymbol{\delta}_Y^i$ *and* κ^i *are given. Then, after some convex conjugate algebra, the separation problem from Theorem 8 reads*

$$\sup \sum_{i=0}^{m} \boldsymbol{a}^i(\boldsymbol{x})^T \boldsymbol{z}^i + (\boldsymbol{b}(\boldsymbol{x}) + \boldsymbol{F}(\boldsymbol{x}) \boldsymbol{\xi})^T \boldsymbol{\lambda} \quad \lambda_0 x_0 \tag{7.22}$$

$$\text{s.t.} \sum_{i=0}^{m} \boldsymbol{K}_Y^{i\,T} \boldsymbol{z}^i + \boldsymbol{\Delta} \boldsymbol{\lambda} + \boldsymbol{\delta}_Y^0 \lambda_0 = \boldsymbol{0} \tag{7.23}$$

$$\left|\left|\boldsymbol{z}^i\right|\right|_{p_i^0} \quad \lambda_i \qquad i = 0, 1, ..., m \tag{7.24}$$

$$\boldsymbol{z}^i \in \mathsf{R}^{n_Y} \qquad i = 0, 1, ..., m \tag{7.25}$$

$$(\lambda_0, \boldsymbol{\lambda}) \in \Lambda \tag{7.26}$$

$$\boldsymbol{\xi} \in \Xi \tag{7.27}$$

where $\boldsymbol{a}^i(\boldsymbol{x}) = \boldsymbol{K}_X^i \boldsymbol{x} + \boldsymbol{\chi}^i$, $\boldsymbol{b}(\boldsymbol{x}) = (\boldsymbol{\delta}_X^{1\,T} \boldsymbol{x} + \kappa^1, \ldots, \boldsymbol{\delta}_X^{m\,T} \boldsymbol{x} + \kappa^m)^T$ *and* $\boldsymbol{\Delta} = (\boldsymbol{\delta}_Y^1, \ldots, \boldsymbol{\delta}_Y^m)$.

Proof. For brevity, the proof is reported in Appendix B. □

Example 12 (Linear case)**.** *Assume that* $g_i(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{t}_{(i)} \boldsymbol{x} + \boldsymbol{w}_{(i)} \boldsymbol{y}$ *for given matrices* $\boldsymbol{w}_{(0)}$, $\boldsymbol{W}$, $\boldsymbol{t}_{(0)}$ *and* $\boldsymbol{T}$*. Then, Theorem 8 yields the following separation problem:*

$$\max \left(\boldsymbol{T}\boldsymbol{x} + \boldsymbol{F}(\boldsymbol{x})\boldsymbol{\xi}\right)^T \boldsymbol{\lambda} + \boldsymbol{t}_{(0)} \boldsymbol{x} \lambda_0 \quad \lambda_0 x_0 \tag{7.28}$$

$$\text{s.t. } \boldsymbol{W}^T \boldsymbol{\lambda} + \boldsymbol{w}_{(0)}^T = \boldsymbol{0} \tag{7.29}$$

$$\boldsymbol{\lambda} \in \Lambda \tag{7.30}$$

$$\boldsymbol{\xi} \in \Xi. \tag{7.31}$$

For the specific case where $\boldsymbol{F}$ *is affine in* $\boldsymbol{x}$, $\boldsymbol{T} = \boldsymbol{0}$ *and* $\boldsymbol{w}_{(0)} = \boldsymbol{0}$*, we enlight that this result is equivalent to Theorem 1 in Ayoub and Poss [2016].*

7.2.2 Generalized Benders Decomposition

In this section, we introduce a new Generalized-Benders-Decomposition algorithm able to solve (7.1) by means of successive separation of infeasible $(x_0, \boldsymbol{x})$ pairs.

For notational convenience, we denote by $\boldsymbol{s}$ a generic tuple $(\boldsymbol{\xi}, \lambda_0, \boldsymbol{\lambda}, \boldsymbol{u}^0, ..., \boldsymbol{u}^m)$, and by S be the set of all such tuples satisfying constraints (7.13)-(7.16). In addition, we introduce function σ defined for each $x_0 \in \mathsf{R}$, $\boldsymbol{x} \in X$ and $\boldsymbol{s} \in S$ as the objective function (7.12), i.e.,

$$\sigma(x_0, \boldsymbol{x}; \boldsymbol{s}) = \quad \sum_{i=0}^{m} \lambda_i g_i \boldsymbol{j}_{\boldsymbol{x}} \left(\frac{\boldsymbol{u}}{\lambda_i}\right) + \boldsymbol{\lambda}^T \boldsymbol{F}(\boldsymbol{x})\boldsymbol{\xi} \quad \lambda_0 x_0.$$

In the following theorem, we use the result from Theorem 8 to introduce an alternative projected formulation of (7.1).

Theorem 9. *Assume* X *is convex, then Problem* (7.1) *is equivalently solved by the following finite-dimensional convex MINLP*

$$\inf x_0 \tag{7.32}$$

$$\text{s.t. } \boldsymbol{x} \in X, x_0 \in \mathsf{R} \tag{7.33}$$

$$\sigma(x_0, \boldsymbol{x}; \boldsymbol{s}) \quad 0 \qquad \forall \boldsymbol{s} \in S. \tag{7.34}$$

Proof. The reformulation holds by Theorem 8. To show that it is convex, we have to show that, for each $\boldsymbol{s} \in S$, function $\sigma(\ ,\ ; \boldsymbol{s})$ is convex with respect to x_0 and $\boldsymbol{x}$. Note that since $\boldsymbol{\lambda}, \boldsymbol{\xi} \quad \boldsymbol{0}$

and function f_{ij} is convex for each $i = 1, \ldots, m$ and $j = 1, \ldots, n_\Xi$, we have that

$$\boldsymbol{x} \mapsto \boldsymbol{\lambda}^T \boldsymbol{F}(\boldsymbol{x})\boldsymbol{\xi} \quad \lambda_0 x_0 \tag{7.35}$$

is convex since the non-negative sum of convex functions is convex and the last term is linear. We therefore focus on the remaining part, and first show that $\boldsymbol{x} \mapsto g_0 j_{\boldsymbol{x}}(\boldsymbol{\pi})$ is a concave function for any fixed $\boldsymbol{\pi} \in \mathbb{R}^{n_Y}$. Thus, let $\boldsymbol{\pi} \in \mathbb{R}^{n_Y}$ be fixed. By definition, we have

$$g_0 j_{\boldsymbol{x}}(\boldsymbol{\pi}) = \sup_{\boldsymbol{y} \in \mathrm{dom}(g_0 j_{\boldsymbol{x}})} \{ \boldsymbol{\pi}^T \boldsymbol{y} \quad g_0 j_{\boldsymbol{x}}(\boldsymbol{y}) \} = \sup_{\boldsymbol{y} \in \mathrm{dom}(g_0 j_{\boldsymbol{x}})} \{ \boldsymbol{\pi}^T \boldsymbol{y} \quad g_0(\boldsymbol{x}, \boldsymbol{y}) \}. \tag{7.36}$$

Let us introduce new variables $\boldsymbol{z} \in \mathbb{R}^{n_X}$ such that $\boldsymbol{z} = \boldsymbol{x}$. Then, the following holds by Lagrangian duality:

$$\begin{aligned}
g_0 j_{\boldsymbol{x}}(\boldsymbol{\pi}) &= \sup_{(\boldsymbol{z},\boldsymbol{y}) \in \mathrm{dom}(g_0), \boldsymbol{z} = \boldsymbol{x}} \{ \boldsymbol{\pi}^T \boldsymbol{y} \quad g_0(\boldsymbol{z}, \boldsymbol{y}) \} && (7.37)\\
&= \inf_{\boldsymbol{\lambda} \in \mathbb{R}^{n_X}} \sup_{(\boldsymbol{z},\boldsymbol{y}) \in \mathrm{dom}(g_0)} \{ \boldsymbol{\lambda}^T (\boldsymbol{z} \quad \boldsymbol{x}) + \boldsymbol{\pi}^T \boldsymbol{y} \quad g_0(\boldsymbol{z}, \boldsymbol{y}) \} && (7.38)\\
&= \inf_{\boldsymbol{\lambda} \in \mathbb{R}^{n_X}} \sup_{(\boldsymbol{z},\boldsymbol{y}) \in \mathrm{dom}(g_0)} \{ \quad \boldsymbol{\lambda}^T \boldsymbol{x} + \begin{pmatrix} \boldsymbol{\lambda} \\ \boldsymbol{\pi} \end{pmatrix}^T \begin{pmatrix} \boldsymbol{z} \\ \boldsymbol{y} \end{pmatrix} \quad g_0(\boldsymbol{z}, \boldsymbol{y}) \} && (7.39)\\
&= \inf_{\boldsymbol{\lambda} \in \mathbb{R}^{n_X}} \{ \quad \boldsymbol{\lambda}^T \boldsymbol{x} + g_0(\boldsymbol{\lambda}, \boldsymbol{\pi}) \}. && (7.40)
\end{aligned}$$

Thus, $g_0 j_{\boldsymbol{x}}(\boldsymbol{\pi})$ can be expressed as the infimum of infitely many affine functions of $\boldsymbol{x}$, thus, it is concave in $\boldsymbol{x}$. Clearly, similar arguments can be applied to each $\boldsymbol{x} \mapsto g_i j_{\boldsymbol{x}}(\boldsymbol{\pi})$ with $i = 1, ..., m$ so as to conclude. □

Algorithm 4: Generalized Benders decomposition

Input: an instance of (7.1), a tolerance ε 0 and an initial set $\hat{S}$ S such that (MP) is bounded.

```
Stop  false ;
while Stop = false do
    Solve (MP) ;
    if (MP) is infeasible then
        Declare (7.1) infeasible and return ;
    Let (x_0, x ) be an optimal solution of (MP) ;
    Solve the separation problem (7.12)-(7.16) for (x_0, x ) ;
    Let s = (ξ , λ_0, λ , u^0 , ..., u^m ) be an optimal solution of the separation problem ;
    if σ(x_0, x ; s ) 0 then
        Ŝ  Ŝ [ {s }
    else
        Stop  true ;
```

Based on Theorem 9, we can derive a cutting-plane algorithm where cuts (7.34) are dynamically generated, for which we discuss finite convergence. The complete procedure is reported in

Algorithm 4, where problem (MP) is defined as,

$$\inf\{x_0 : (x_0, \boldsymbol{x}) \in \mathbb{R} \quad X \quad \sigma(x_0, \boldsymbol{x}; \boldsymbol{s}) \quad 0 \ \forall \boldsymbol{s} \in \hat{S}\} \tag{MP}$$

for some $\hat{S}$ S which is dynamically augmented.

7.2.3 Column-and-constraint generation

As a second solution approach, we generalize the column-and-constraint algorithm introduced in Zeng and Zhao [2013] to our convex setting. A complete description of the algorithm is given in Algorithm 5, where $(\widetilde{MP})$ is defined as,

$$\inf\{x_0 : (x_0, \boldsymbol{x}) \in \mathbb{R} \quad X \quad \boldsymbol{y}_{\hat{\boldsymbol{\xi}}} \in Y(\boldsymbol{x}, \hat{\boldsymbol{\xi}}) \wedge x_0 \quad g_0(\boldsymbol{x}, \boldsymbol{y}_{\hat{\boldsymbol{\xi}}}) \ \forall \hat{\boldsymbol{\xi}} \in \hat{\Xi}\} \tag{$\widetilde{MP}$}$$

for some $\hat{\Xi}$ Ξ which is dynamically augmented.

Algorithm 5: Column-and-constraint generation

Input: an instance of (7.1) and an initial set $\hat{\Xi}$ Ξ such that $(\widetilde{MP})$ is bounded.

$HasConverged$ $false$;
while $HasConverged = false$ **do**
  Solve $(\widetilde{MP})$;
  if $(\widetilde{MP})$ *is infeasible* **then**
    Declare (7.1) infeasible and **stop** ;
  Let $(x_0, \boldsymbol{x}\ )$ be an optimal solution of $(\widetilde{MP})$;
  Solve the separation problem ;
  (7.12)-(7.16) for $(x_0, \boldsymbol{x}\ )$;
  Let $\boldsymbol{s}\ = (\boldsymbol{\xi}\ , \lambda_0, \boldsymbol{\lambda}\ , \boldsymbol{u}^0\ , ..., \boldsymbol{u}^m\ )$ be an optimal solution of the separation problem ;
  if $\sigma(x_0, \boldsymbol{x}\ ; \boldsymbol{s}\ ) > 0$ **then**
    $\hat{\Xi}$ $\hat{\Xi} \cup \{\boldsymbol{\xi}\}$;
  else
    $HasConverged$ $true$;

7.2.4 Convergence

In this section, we study the convergence properties of the generalized Benders decomposition and the column-and-constraint generation algorithm. To ease our discussion, we first introduce some definitions.

Definition 3 (ε-oracle for separation)**.** *Let* ε 0*. We say that an oracle is an* ε*-oracle for the separation problem if and only if, given pair* $(x_0, \boldsymbol{x}) \in \mathbb{R}$ X*, it returns* $\tilde{\boldsymbol{s}}$ *such that*

1. $\tilde{\boldsymbol{s}} \in S$;
2. $\sup_{\boldsymbol{s} \in S} \sigma(x_0, \boldsymbol{x}; \boldsymbol{s})$ $\sigma(x_0, \boldsymbol{x}, \tilde{\boldsymbol{s}})$ ε.

Moreover, we say that the ε*-oracle is* stable *if for a given* $(x_0, \boldsymbol{x})$*, different calls to the oracle yield the same* $\tilde{\boldsymbol{s}}$*. It is called* extreme *if the returned* $\tilde{\boldsymbol{s}} = (\boldsymbol{\xi}, \lambda_0, \boldsymbol{\lambda}, \boldsymbol{u}^0, ..., \boldsymbol{u}^m)$ *is such that* $\boldsymbol{\xi} \in \text{vert}\,(\Xi)$*.*

Definition 4 (ε-relaxed feasible space)**.** *We de ne the ε-relaxed feasible space as*

$$Y^{\varepsilon}(\boldsymbol{x},\boldsymbol{\xi}) = \{\boldsymbol{y} \in \mathbb{R}^{n_Y} : \boldsymbol{F}(\boldsymbol{x})\boldsymbol{\xi} + \boldsymbol{g}(\boldsymbol{x},\boldsymbol{y}) \leq \boldsymbol{\varepsilon}\boldsymbol{e}\}. \tag{7.41}$$

Moreover, we de ne the ε-relaxed version of (7.1) *as the problem obtained by substituting $Y(\boldsymbol{x},\boldsymbol{\xi})$ by $Y^{\varepsilon}(\boldsymbol{x},\boldsymbol{\xi})$ in* (7.1)*.*

We are now ready to state our first convergence result regarding the generalized Benders decomposition algorithm.

Theorem 10. *Assume solving the separation problem with a stable ε-oracle. Then, Algorithm 4 nitely terminates. Moreover, it either returns a solution $(x_0,\boldsymbol{x})$ such that $\forall \boldsymbol{\xi} \in \Xi, \exists \boldsymbol{y} \in Y^{\varepsilon}(\boldsymbol{x},\boldsymbol{\xi}), x_0 \geq g_0(\boldsymbol{x},\boldsymbol{y}) - \varepsilon$, or correctly concludes that* (7.1) *is infeasible.*

Proof.

1. First, observe that the separation problem from Theorem 8 is an uncoupled biconvex problem. Thus, at most $|\mathrm{vert}(\Xi)|$ cuts of type (7.34) exist. We claim that no first-stage solution can repeat itself so that at most $|\mathrm{vert}(\Xi)|+1$ iterations are needed to terminate. Assume, by contradiction, that $(\hat{x}_0,\hat{\boldsymbol{x}})$ is repeated, meaning that the cut "$\sigma(x_0,\boldsymbol{x};\hat{\boldsymbol{s}}) \geq 0$" is part of MP at the second apparition of $(\hat{x}_0,\hat{\boldsymbol{x}})$, $\hat{\boldsymbol{s}}$ being the solution returned by the oracle for $(\hat{x}_0,\hat{\boldsymbol{x}})$. By feasibility, it holds $\sigma(\hat{x}_0,\hat{\boldsymbol{x}};\hat{\boldsymbol{s}}) \leq 0$. This implies that $\sup_{\boldsymbol{s}\in S}\sigma(\hat{x}_0,\hat{\boldsymbol{x}};\boldsymbol{s}) \leq \varepsilon$. This, in turn, implies the termination of the algorithm. (Here, we used the fact that the ε-oracle is stable.)

2. Assume that, at some iteration, the oracle returns a solution $\boldsymbol{s}$ such that $\sigma(x_0,\boldsymbol{x};\boldsymbol{s}) \geq 0$. Together with $\sup_{\boldsymbol{s}\in S}\sigma(x_0,\boldsymbol{x};\boldsymbol{s}) \geq \sigma(x_0,\boldsymbol{x},\boldsymbol{s}) - \varepsilon$, we have that $\sup_{\boldsymbol{s}\in S}\sigma(x_0,\boldsymbol{x};\boldsymbol{s}) \geq -\varepsilon$. In turn, this implies

$$\begin{aligned}
&\sup_{\boldsymbol{s}\in S}\sigma(x_0,\boldsymbol{x};\boldsymbol{s}) && (7.42)\\
=&\sup_{\boldsymbol{\xi}\in\Xi,(\lambda_0,\boldsymbol{\lambda})\in\Lambda}\inf_{\boldsymbol{y}\in\mathbb{R}^{n_Y}}\left\{\boldsymbol{\lambda}^T(\boldsymbol{F}(\boldsymbol{x})\boldsymbol{\xi}+\boldsymbol{g}(\boldsymbol{x},\boldsymbol{y}))+\lambda_0(g_0(\boldsymbol{x},\boldsymbol{y}) - x_0)\right\} && (7.43)\\
=&\sup_{\boldsymbol{\xi}\in\Xi}\inf_{\boldsymbol{y}\in\mathbb{R}^{n_Y}}\sup_{(\lambda_0,\boldsymbol{\lambda})\in\Lambda}\left\{\boldsymbol{\lambda}^T(\boldsymbol{F}(\boldsymbol{x})\boldsymbol{\xi}+\boldsymbol{g}(\boldsymbol{x},\boldsymbol{y}))+\lambda_0(g_0(\boldsymbol{x},\boldsymbol{y}) - x_0)\right\} && (7.44)\\
=&\sup_{\boldsymbol{\xi}\in\Xi}\inf_{\boldsymbol{y}\in\mathbb{R}^{n_Y}}\max\left\{\max_{i=1,\dots,m}\left\{\boldsymbol{f}_{(i)}(\boldsymbol{x})\boldsymbol{\xi}+g_i(\boldsymbol{x},\boldsymbol{y})\right\}; g_0(\boldsymbol{x},\boldsymbol{y}) - x_0; 0\right\} && (7.45)\\
&\leq \varepsilon. && (7.46)
\end{aligned}$$

   Here, the minimax theorem from Perchet and Vigeral [2015] was used to swap the sup and inf operators. This shows that $(x_0,\boldsymbol{x})$ is such that $\forall \boldsymbol{\xi} \in \Xi, \exists \boldsymbol{y} \in Y^{\varepsilon}(\boldsymbol{x},\boldsymbol{\xi}), x_0 \geq g_0(\boldsymbol{x},\boldsymbol{y}) - \varepsilon$.

3. Assume that, at some iteration, (MP) is infeasible. This means that

$$\forall (x_0,\boldsymbol{x}) \in \mathbb{R} \times X, \exists \hat{\boldsymbol{s}} \in \hat{S}, \quad \sigma(x_0,\boldsymbol{x};\hat{\boldsymbol{s}}) > 0 \tag{7.47}$$

   which implies that

$$\forall (x_0,\boldsymbol{x}) \in \mathbb{R} \times X, \quad \sup_{\boldsymbol{s}\in S}\sigma(x_0,\boldsymbol{x};\boldsymbol{s}) > 0. \tag{7.48}$$

By Theorem 8, this shows that (7.1) is infeasible.

□

This theorem directly implies convergence of the column-and-constraint generation algorithm, as announced in the following theorem.

Theorem 11. *Assume solving the separation problem with an extreme ε-oracle. Then, Algorithm 4 nitely terminates. Moreover, for a su ciently small $\varepsilon > 0$, it either returns a solution $(x_0, \boldsymbol{x})$ such that*

{ $8\boldsymbol{\xi} \in \hat{\Xi}, 9\boldsymbol{y} \in Y(\boldsymbol{x}, \boldsymbol{\xi}), x_0 \quad g_0(\boldsymbol{x}, \boldsymbol{y})$;

{ $8\boldsymbol{\xi} \in \Xi \backslash \hat{\Xi}, 9\boldsymbol{y} \in Y^{\varepsilon}(\boldsymbol{x}, \boldsymbol{\xi}), x_0 \quad g_0(\boldsymbol{x}, \boldsymbol{y}) \quad \varepsilon$,

or correctly concludes that (7.1) is infeasible. The result also holds for $\varepsilon = 0$.

Proof. The proof is similar to that of the previous theorem. The stronger characterization of the returned solution directly comes from the fact that the oracle is extreme and the associated constraints are exact. □

7.2.5 0-1 polytopic uncertainty sets

Algorithms 4 and 5 give general schemes for solving problem (7.1), though its applicability depends on the possibility of solving the separation problem in practice. In this section, we present a convex MINLP formulation of the latter in the relevant case in which Ξ is as an affine mapping of a 0-1 polytope. Note that, given any polytopic uncertain set, there exists an affine mapping to a 0-1 polytope. To see this, one can always express Ξ as the set of convex combinations of its extreme points. In the following, we will denote by $\Omega \quad \mathsf{R}^{n_\Omega}$ the 0-1 polytope associated with Ξ, and assume its dimension n_Ω be manageable. Clearly, when $\Xi \quad [0,1]^{n_\Xi}$, $\boldsymbol{U}$ is a totally unimodular matrix and $\boldsymbol{d}$ is integral (see (7.2)), the identity mapping can be used and $n_\Omega = n_\Xi$. This is notably the case for the budgeted uncertainty set (see Bertsimas and Sim [2004]) with an integer parameter. For the case where the budget parameter is fractional, Ayoub and Poss [2016] shows that an affine mapping with a 0-1 polytope of size $2n_\Xi$ exists. We now state our theorem.

Theorem 12. *Let $\Omega \quad \mathsf{R}^{n_\Omega}$ be a given 0-1 polytope and let $\boldsymbol{\rho}^0, \boldsymbol{\rho}^1, ..., \boldsymbol{\rho}^{n_\Omega} \in \mathsf{R}^{n_\Xi}$ be some vectors. Assume that $\Xi = \tilde{\boldsymbol{\rho}}(\Omega)$ where $\tilde{\boldsymbol{\rho}} : \boldsymbol{\omega} \mapsto \boldsymbol{\rho}^0 + \sum_{k=1}^{n_\Omega} \boldsymbol{\rho}^k \omega_k$. Then, given a pair $(x_0, \boldsymbol{x}) \in \mathsf{R} \quad X$, the separation model introduced in Theorem 8 can be reformulated as the following convex MINLP.*

$$\sup \quad \sum_{i=0}^{m} \lambda_i g_i \boldsymbol{j}_{\boldsymbol{x}} \left(\frac{\boldsymbol{u}}{\lambda_i} \right) + \boldsymbol{\lambda}^T \boldsymbol{F}(\boldsymbol{x}) \boldsymbol{\rho}^0 + \sum_{k=1}^{n_\Omega} \boldsymbol{\theta}^{k^T} \boldsymbol{F}(\boldsymbol{x}) \boldsymbol{\rho}^k \quad \lambda_0 x_0 \tag{7.49}$$

$$\text{s.t.} \sum_{i=0}^{m} \boldsymbol{u}^i = \boldsymbol{0} \tag{7.50}$$

$$(\lambda_0, \boldsymbol{\lambda}) \in \Lambda \tag{7.51}$$

$$\boldsymbol{\theta}^k \quad \boldsymbol{\lambda} \qquad k = 1, ..., n_\Omega \tag{7.52}$$

$$\boldsymbol{\theta}^k \quad \omega_k \boldsymbol{e} \qquad k = 1, ..., n_\Omega \tag{7.53}$$

$$\boldsymbol{\theta}^k \quad \boldsymbol{\lambda} + \omega_k \boldsymbol{e} \quad \boldsymbol{e} \qquad k = 1, ..., n_\Omega \tag{7.54}$$

$$\boldsymbol{\theta}^k \; 2 \; \mathsf{R}_+^m \qquad k = 1, ..., n_\Omega \tag{7.55}$$

$$\boldsymbol{\omega} \; 2 \; \Omega \setminus f0, 1g^{n_\Omega} \tag{7.56}$$

$$\boldsymbol{u}^i \; 2 \; \mathsf{R}^{n_Y} \qquad i = 0, 1, ..., m \tag{7.57}$$

Proof. By replacing each $\boldsymbol{\xi} \; 2 \; \Xi$ by an $\boldsymbol{\omega} \; 2 \; \Omega$ such that $\boldsymbol{\xi} = \boldsymbol{\rho}^0 + \sum_{k=1}^{n_\Omega} \boldsymbol{\rho}^k \omega_k$, objective function (7.12) can be rewritten as

$$\sup \quad \sum_{i=0}^{m} \lambda_i g_i \boldsymbol{j}_{\boldsymbol{x}} \left(\frac{\boldsymbol{u}^i}{\lambda_i} \right) + \boldsymbol{\lambda}^T \boldsymbol{F}(\boldsymbol{x}) \boldsymbol{\rho}^0 + \sum_{k=1}^{n_\Omega} \boldsymbol{\lambda}^T \boldsymbol{F}(\boldsymbol{x}) \boldsymbol{\rho}^k w_k \quad \lambda_0 x_0$$

Noting that this function includes bilinear terms, we can restrict our attention to $\omega \; 2 \; \text{vert}\,(\Omega)$ $f0, 1g^{n_\Omega}$. By introducing variables $\theta_i^k = \lambda_i \omega_k$ ($i = 1, ..., m$ and $k = 1, ..., n_\Omega$), the bilinear term can be linearized as follows

$$\sum_{k=1}^{n_\Omega} \boldsymbol{\lambda}^T \boldsymbol{F}(\boldsymbol{x}) \boldsymbol{\rho}^k w_k = \sum_{k=1}^{n_\Omega} \sum_{i=1}^{m} \sum_{j=1}^{n_\Xi} f_{ij}(\boldsymbol{x}) \rho_j^k \underbrace{\lambda_i \omega_k}_{=\theta_i^k} = \sum_{k=1}^{n_\Omega} {\boldsymbol{\theta}^k}^T \boldsymbol{F}(\boldsymbol{x}) \boldsymbol{\rho}^k. \tag{7.58}$$

The result follows as (7.52)–(7.54) are linearization constraints and $0 \quad \lambda_i \quad 1$ by assumption. □

7.3 Application: resource planning problem

We tested our method on a stochastic resource planning problem from the literature, modified so as to incorporate resource congestion.

Our algorithm was implemented in C++17 using Gurobi 9.5 to solve the underlying optimization problems. All experiments were conducted on an AMD Ryzen 5 PRO 4650GE at 3.3 GHz, with a time limit equal to 3,600 CPU seconds per run.

7.3.1 Problem description

We are given a set I of resources (e.g., server types) and a set J of customers. Each resource $i \; 2 \; I$ is associated to a unitary cost c_i, and each customer $j \; 2 \; J$ has a demand d_j. We denote by μ_{ij} the service rate of resource i for customer j, i.e., the fraction of demand of j can be served by a unit of i. The deterministic problem introduced in Luedtke [2010] asks to serve all customers' demands while minimizing the total cost of the used resources, and is formulated as follows

$$\min \sum_{i \, 2 \, I} c_i x_i \tag{7.59}$$

$$\text{s.t.} \sum_{j \, 2 \, J} y_{ij} \quad x_i \qquad\qquad i \; 2 \; I \tag{7.60}$$

$$\sum_{i \, 2 \, I} \mu_{ij} y_{ij} \quad d_j \qquad\qquad j \; 2 \; J \tag{7.61}$$

$$x_i \quad 0 \qquad\qquad i \; 2 \; I \tag{7.62}$$

$$y_{ij} \quad 0 \qquad\qquad (i, j) \; 2 \; I \quad J, \tag{7.63}$$

Here, each variable x_i ($i \in I$) represents the amount of resource i to buy, while y_{ij} ($(i,j) \in I \times J$) is the amount of resource i allocated to customer j.

We modify the model to consider congestion, which may reduce the possibility for the customers to access the resources, thus requiring an increased amount of resource to guarantee a given service rate. Accordingly, constraints (7.60) are replaced by

$$\Big(1 + \alpha_i \big[\sum_{j \in J} y_{ij}\big]^{\beta_i}\Big)\Big(\sum_{j \in J} y_{ij}\Big) \leq x_i \qquad i \in I, \tag{7.64}$$

where α_i and β_i are given parameters related to resource $i \in I$. This congestion function, which is convex for any non-negative α_i and β_i, was introduced by Desrochers et al. [1995] for a facility location problem.

In a two-stage setting, customers' demands are typically not known when the resources have to be bought, whereas the assignment of customers to resources can be postponed after the actual realization of the demands reveals. We assume that the demand of each customer j has a nominal value $\bar{d}_j$ and maximum deviation $\tilde{d}_j$. The resulting demand is $d_j = \bar{d}_j + \xi_j \tilde{d}_j$, where ξ_j is a random parameter and $\boldsymbol{\xi}$ belongs to a budgeted uncertainty set $\Xi = \{\boldsymbol{\xi} \in [0,1]^{|J|} : \sum_{j \in J} \xi_j \leq \Gamma\}$.

7.3.2 Instance generation

We generated random instances as follows: for each resource i and customer j, the service rate μ_{ij} is defined as $U(0,1)$, where $U(a,b)$ denotes the uniform distribution between a and b. For each customer j, the nominal demand $\bar{d}_j$ is taken from $U(1,50)$, and the maximum deviation $\tilde{d}_j$ is either $0.05\bar{d}_j$ or $0.10\bar{d}_j$. For each resource i, the cost c_i is in $U(8,10)\sum_{j \in J} \mu_{ij}/|J|$, so that the higher the average efficiency the more costly the resource. In addition, congestion of the resource is defined by α_i in $U(0,1)$ and $\beta_i = 1$.

We generated instances of different sizes ($|I|, |J|$) equal to $(10,20)$, $(10,30)$, $(15,30)$, $(15,40)$, $(20,40)$ and $(20,50)$. For each size, 5 instances were generated. Finally, the uncertainty budget Γ was set to $\lfloor p|J| \rfloor$ where $p \in \{0.05, 0.10, 0.20\}$, i.e., up to $p\%$ of the clients change their demands to the maximum value.

7.3.3 Results

Table 7.1 gives the outcome of our computational experiments and reports, for each group of 5 homogeneous instances, the number of cases in which the algorithm computes a provably optimal solution, and the corresponding average computing time and average number of times the separation problem was solved.

The results show that our approach is able to solve to optimality all instances with $p = 0.05$, with an average computing time raising from few seconds to almost 15 minutes. Increasing the value of p yields more challenging instances. This is shown both by the number of optimal solution (51 out of 60) and by the average computing time for solved instances, which can be as large as twice the time needed in the previous case. This trend is more evident for $p = 0.20$, in which case the algorithm solves to optimality only 36 instances (out of 60) with a significantly larger computing time. Concerning the number of separation rounds, it grows with the size of the instance, whereas the value of p is immaterial. Finally, instances with $\bar{d}_j/\tilde{d}_j = 0.05$ appear to be

			$p = 0.05$			$p = 0.10$			$p = 0.20$		
jIj	jJj	$\bar{d}_j/\tilde{d}_j$	opt	time	$j\hat{S}j$	opt	time	$j\hat{S}j$	opt	time	$j\hat{S}j$
10	20	0.05	5	13.92	187.00	5	15.82	201.00	5	19.63	201.40
		0.10	5	15.13	194.60	5	16.39	197.40	5	21.05	202.40
	30	0.05	5	23.41	213.80	5	35.76	216.80	5	386.51	222.00
		0.10	5	23.81	207.80	5	45.37	223.20	5	290.78	223.40
15	30	0.05	5	64.01	406.60	5	118.86	412.00	4	517.51	451.00
		0.10	5	70.41	429.20	5	175.26	426.00	5	357.54	420.60
	40	0.05	5	129.24	381.60	3	1133.50	346.00	1	215.05	311.00
		0.10	5	143.41	387.40	5	743.03	382.60	1	2507.04	313.00
20	40	0.05	5	383.06	664.40	3	1900.71	675.33	0	—	—
		0.10	5	455.72	669.00	5	847.54	695.40	3	952.38	669.67
	50	0.05	5	785.16	692.60	1	801.34	532.00	0	—	—
		0.10	5	853.19	683.60	4	1507.89	646.25	2	3117.22	692.00

Table 7.1: Computational results on the resource allocation problem with different uncertainty budgets.

more challenging for our algorithm.

7.4 Conclusion

In this Chapter, we studied general adjustable robust problems where the second-stage feasible space is defined by means of convex constraints. Using Fenchel duality, we were able to derive a nonconvex separation problem leading to a generalization of well-known algorithmic methods for linear adjustable robust optimization. We also showed how a special structure of the uncertainty set can be exploited to derive a convex MINLP formulation of the separation problem. Thus allowing the practical implementation of the aforementioned algorithms to solve real-world problems. We confirmed this by reporting computational experiments on a planning problem which we were able to solve to optimality for reasonable sizes.

Conclusion

This chapter concludes this thesis by recalling its main contributions to the robust optimization community and by suggesting future research directions.

Recall that the present thesis was concerned by optimization problems suffering from imprecise input data and featuring a two-stage decision process. While different approaches exist in the scientific literature (see Chapter 1), the robust optimization approach was the one employed in this research. This was motivated by the weaker assumptions (compared to other approaches) which it imposes. These weaker assumptions allow for a wider applicability in real-world applications.

Unfortunately, we also showed, in Chapter 2, that the two-stage RO approach led to hard optimization problems. In particular, we made clear that further effort was (and still is) needed to tackle the large class of problems where feasible second-stage decisions are describded by nonlinear constraints (in which we include integrality requirements).

In the next section, we recall the main contributions which were presented throughout this thesis.

7.5 Main contributions

7.5.1 Mixed-integer second stage

A large part of this thesis was dedicated to problems where second-stage decisions have to take integer values. For this class of problems, only special cases were treated in the scientific literature. We therefore tried to contribute to the current state of the art by introducing two new theoretical results and to apply them to two real-world problems.

Objective uncertainty

The first line of research has been on objective uncertain two-stage problems. For this class of problem, a decomposition scheme had been introduced for the special case where constraints are linear and the constraints linking the first- and second-stage decisions have special structure.

In Chapter 3, we showed how this approach could be generalized to a much broader class of problems where the second-stage is defined by general convex constraints and removing any

assumption on the linking constraints between the first and second stage. We enlight that this generalization required further analysis and is not a straightforward extension of the original approach. In particular, it requires the use of Fenchel duality, spatial branching and asymptotic convergence.

In Chapter 4, we applied the previous method to a scheduling problem where one has to select of set of jobs to schedule on a single machine while the output of each task may be deteriorated by some random event. We showed that our approach was actually faster in solving this problem compared to existing solution schemes which, contrary to our method, only report heuristic solutions.

Binary uncertainty

Chapter 5 studied ARO problems where the second stage is defined by mixed-integer constraints under the specific case where the uncertainty is discrete (in fact, binary). In this setting, no previous work could be found for a setting as general as the one our work addresses. We showed how any such problem with constraint uncertainty can be reformulated as a problem with objective uncertainty, allowing us to derive a computationally sound method for solving this class of problems.

In Chapter 6, we applied the previous method to a facility location problem, a famous optimization problem arising in logistics, where clients' demands are uncertain. We showed that our method could solve instances of reasonable size.

7.5.2 Convex second stage

In the last chapter of this thesis (i.e., Chapter 7), we studied ARO problems where the second-stage decisions are defined by means of convex constraints. In this setting, no exact approaches could be found for the general case. We showed that two approaches developed for the linear setting could be extended to the convex setting by using Fenchel duality and exploiting a special structure of the uncertainty set. Unfortunately, we also showed that this generalization has weaker guarantees than their linear counterparts in terms of solution quality.

7.6 Future research directions

We end by discussing future research directions.

7.6.1 Exploiting structure from deterministic problems

While more and more effort is done in deriving exact methods for solving two-stage robust problems with mixed-integer second stage, the existing algorithms are often very sensitive to the size of the second-stage problem (in terms of number of constraints and/or variables). This remains true when the deterministic problem has a "nice" structure (in terms of decomposition). For instance, for a large class of problems, Dantzig-Wolfe or Benders decomposition have been specialized to obtain very efficient methods. Unfortunately, the same cannot be said regarding the robust counterpart of such problems. A potential line of research could be to develop solution schemes which preserve

the deterministic problem's structure. A first result was introduced by Kämmerling and Kurtz [2020] for problems with objective uncertainty.

7.6.2 Improving solution guarantees for problems with second-stage problems

While the Benders decomposition and the column-and-constraint generation algorithm were generalized in Chapter 7 to problems with convex second stage, the quality of the returned solution may not be satisfactory for some critical applications. Another line of research could be to try to alleviate these limitations by developing new algorithmic methods. Informally speaking, it could actually be that this research direction and the previous are connected, if not equivalent.

Appendices

APPENDIX A

Recalls on convex optimization

A.1 Fenchel duality

A.1.1 Introduction

A dual perception

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a closed convex function over X. A classical representation for f is to consider a set of values $P = \{(\boldsymbol{x}, f(\boldsymbol{x})) : \boldsymbol{x} \in X\}$. Yet, it is not the only way to represent f. Indeed, by convexity, f can be represented by a set of infinitely many "linear approximating hyperplanes" (in fact, supporting hyperplanes of the epigraph of f). Indeed, considering a point $\boldsymbol{x}_0 \in \mathbb{R}^n$, there exists $\boldsymbol{\pi} \in \mathbb{R}^n$ and $b \in \mathbb{R}$ such that,

$$\boldsymbol{\pi}^T \boldsymbol{x}_0 - b = f(\boldsymbol{x}_0) \qquad \text{and} \qquad \boldsymbol{\pi}^T \boldsymbol{x} - b \leq f(\boldsymbol{x}), \forall \boldsymbol{x} \in \mathbb{R}^n.$$

Note that these equations provide, knowing only $\boldsymbol{\pi}$ (i.e., the slope of a hyper-plane), a way to compute b (i.e., the y-intercept of the hyper-plane). Indeed, the following holds:

$$b = \sup_{\boldsymbol{x} \in \mathbb{R}^n} \left\{ \boldsymbol{\pi}^T \boldsymbol{x} - f(\boldsymbol{x}) \right\}$$

Note that the optimal solution $\boldsymbol{x}$ is exactly $\boldsymbol{x}_0$. Forgetting about point $\boldsymbol{x}_0$, these small considerations show that, given any slope for a supporting hyper-plane $\boldsymbol{\pi}$, one is able to compute its offset. This offset is then noted $f^*(\boldsymbol{\pi})$ and function f^* is called the convex conjugate of f. Moreover, because f is equivalently described by its linear approximating hyperplanes, one obtains a so-called *dual* representation of f, given as $D = \{(\boldsymbol{\pi}, f^*(\boldsymbol{\pi})) : \boldsymbol{\pi} \in \mathbb{R}^n\}$.

In the following table, we detail the dual representation of a function f.

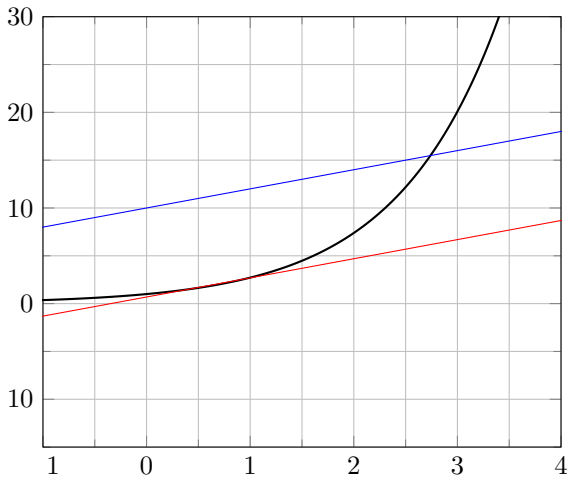


Figure A.1: Geometric interpretation of the convex conjugate for $f(x) = e^x$ and $\pi = 2$, $f^*(2) = \ln(2)$

	Primal	(Fenchel) Dual
Type	points	hyperplanes
Variables	$\boldsymbol{x} \in \mathbb{R}^n$	$\boldsymbol{\pi} \in \mathbb{R}^n$
Values	$f(\boldsymbol{x})$	$f^*(\boldsymbol{\pi})$

In figure A.1, the convex function $f : x \mapsto e^x$ is drawn. A hyper-plane with slope $\pi = 2$ is also shown in blue. In red, a supporting hyper-plane of f with slope $\boldsymbol{\pi}$ is depicted. Its y-intercept is $f^*(2) = \ln(2)$.

An economic interpretation

We briefly give here an economic interpretation of the convex conjugate. Assume that $f(x)$ represents the cost for producing x products. Now, let π be the market price per unit for this product.

Assuming that every item is sold, the generated income is given by πx (unitary price number of goods). The profit corresponds to the income minus the production price $f(x)$. Thus, the profit is given by $\pi x - f(x)$. In search for the maximum profit one can make, we have to solve the following:

$$\sup_{x \in \mathrm{dom}(f)} \{\pi x - f(x)\}. \tag{A.1}$$

Thus $f^*(\pi)$ is exactly the maximum profit one can make when the market price is π.

A relative in mum interpretation

Consider a function f, its infimum is given by the following.

$$\inf_{\boldsymbol{x} \in \mathrm{dom}(f)} f(\boldsymbol{x}) = -\sup_{\boldsymbol{x} \in \mathrm{dom}(f)} \{0 - f(\boldsymbol{x})\} \tag{A.2}$$

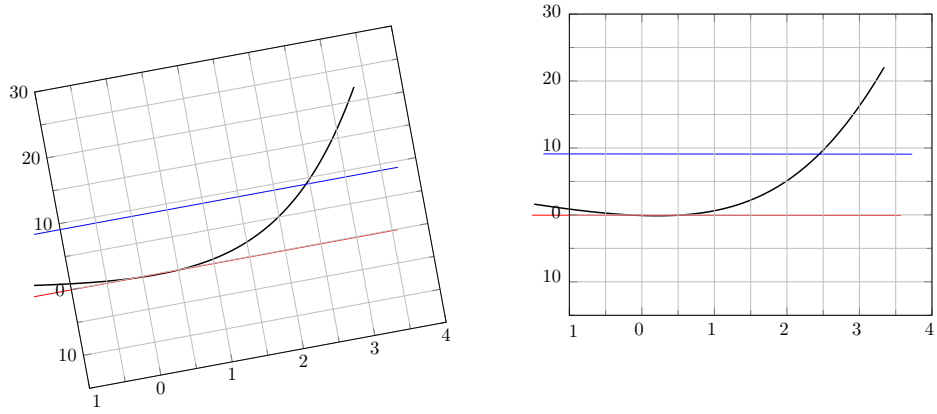


Figure A.2: $f\ (\pi)$ can also be seen as the minimum of f with respect to the axis $y = \pi x$

Yet, what if one wants to find the infimum of f with respect to a given hyperplane, say, $0 = \boldsymbol{\pi}^T \boldsymbol{x}$? This can be done as follows:

$$\sup_{x 2 \mathrm{dom}(f)} f\pi x \quad f(x) g \tag{A.3}$$

Thus, $f\ (\boldsymbol{\pi})$ is also the infimum of function f with respect to the hyper-plane $0 = \boldsymbol{\pi}^T \boldsymbol{x}$.

In figure A.2, we rotated the axes so that the x-axis may be parallel to the blue line $(y = 2x)$. The minimum of the "rotated f" is attained at $f\ (2)$.

A.1.2 Examples

Example 13 (Affine functions)**.** *Assume $f(\boldsymbol{x}) = \boldsymbol{a}^T \boldsymbol{x} + \boldsymbol{b}$, then,*

$$f\ (\boldsymbol{y}) = \begin{cases} \boldsymbol{b} & \text{if } \boldsymbol{y} = \boldsymbol{a} \\ +1 & \text{otherwise} \end{cases}$$

Example 14 (Convex quadratic functions)**.** *Assume $f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}^T \boldsymbol{x} + c$ with $\boldsymbol{A}$ a psd matrix, then,*

$$f\ (\boldsymbol{y}) = \begin{cases} \frac{1}{2}(\boldsymbol{y} \quad \boldsymbol{b})^T \boldsymbol{A}^{\dagger}(\boldsymbol{y} \quad \boldsymbol{b}) \quad c & \text{if } \boldsymbol{y}\ 2\ \mathrm{span}(\boldsymbol{A}) + \boldsymbol{b} \\ +1 & \text{otherwise} \end{cases}$$

where $\boldsymbol{A}^{\dagger}$ is the Moore-Penrose pseudo inverse of $\boldsymbol{A}$. If f is strictly convex (i.e., $\boldsymbol{A}$ is positive de nite), then $\boldsymbol{A}^{\dagger} = \boldsymbol{A}^{\ 1}$ and the column span of $\boldsymbol{A}$ is $\mathbb{R}^n$.

Example 15 (Maximum)**.** *Assume $f(\boldsymbol{x}) = \max_{i=1,\ldots,n} x_i$, then,*

$$f\ (\boldsymbol{y}) = \begin{cases} 0 & \text{if } \sum_{i=1}^{n} y_i = 1, \boldsymbol{y} \quad \boldsymbol{0} \\ +1 & \text{otherwise} \end{cases}$$

Example 16 (Soft max.). *Assume* $f(\boldsymbol{x}) = \log\left(\sum_{i=1}^{n} e^{x_i}\right)$, *then,*

$$f^{*}(\boldsymbol{y}) = \begin{cases} \sum_{i=1}^{n} y_i \log(y_i) & \text{if } \sum_{i=1}^{n} y_i = 1, \boldsymbol{y} \quad \boldsymbol{0} \\ +\infty & \text{otherwise} \end{cases}$$

Example 17 (Norms). *Assume* $f(\boldsymbol{x}) = \|\boldsymbol{x}\|$, *then,*

$$f^{*}(\boldsymbol{y}) = \begin{cases} 0 & \text{if } \|\boldsymbol{y}\|^{*} \quad 1 \\ +\infty & \text{otherwise} \end{cases}$$

where $\|\cdot\|^{*}$ *is the dual norm of* $\|\cdot\|$.

Example 18 (Negative entropy). *Assume* $f(\boldsymbol{x}) = \sum_{i=1}^{n} x_i \log x_i$ *with* $\mathrm{dom}(f) = \mathbb{R}^{n}_{++}$, *then,*

$$f^{*}(\boldsymbol{y}) = \sum_{i=1}^{n} e^{y_i - 1}$$

A.1.3 Calculus rules

Proposition 6 (Addition to affine mapping). *Assume* $f(\boldsymbol{x}) = \tilde{f}(\boldsymbol{x}) + \boldsymbol{a}^{T}\boldsymbol{x} + b$, *then,*

$$f^{*}(\boldsymbol{y}) = \tilde{f}^{*}(\boldsymbol{y} \quad \boldsymbol{a}) \quad b$$

Proposition 7 (Composition with an invertible affine mapping). *Assume* $f(\boldsymbol{x}) = \tilde{f}(\boldsymbol{A}\boldsymbol{x}+\boldsymbol{b})$ *with* $\det(\boldsymbol{A}) \neq 0$, *then,*

$$f^{*}(\boldsymbol{y}) = \tilde{f}^{*}\left(\boldsymbol{A}^{-T}\boldsymbol{y}\right) \quad \boldsymbol{b}^{T}\boldsymbol{A}^{-T}\boldsymbol{y}$$

Proposition 8 (Separable sums). *Assume* $f(\boldsymbol{x}_1, \boldsymbol{x}_2) = f_1(\boldsymbol{x}_i) + f_2(\boldsymbol{x}_2)$, *then,*

$$f^{*}(\boldsymbol{y}_1, \boldsymbol{y}_2) = f_1^{*}(\boldsymbol{y}_1) + f_2^{*}(\boldsymbol{y}_2)$$

Proposition 9 (Non-separable sums). *Assume* $f(\boldsymbol{x}) = \sum_{i=1}^{p} f_i(\boldsymbol{x})$, *then,*

$$\begin{aligned} f^{*}(\boldsymbol{y}) = \quad & \inf \quad && \textstyle\sum_{i=1}^{p} f_i^{*}\left(\boldsymbol{v}^{(i)}\right) \\ & \text{s.t.} \quad && \textstyle\sum_{i=1}^{p} \boldsymbol{v}^{(i)} = \boldsymbol{y} \\ & && \boldsymbol{V} \in \mathbb{R}^{p \quad n} \end{aligned}$$

Proposition 10 (Scalar multiplication). *Assume* $f(\boldsymbol{x}) = \alpha\tilde{f}(\boldsymbol{x})$, *then,*

$$f^{*}(\boldsymbol{y}) = \alpha\tilde{f}^{*}\left(\frac{\boldsymbol{y}}{\alpha}\right)$$

Proposition 11 (Convex/Concave conjugate). *Let* f *be a given function, then* $(-f)^{*}(\boldsymbol{y}) = -f^{*}(-\boldsymbol{y})$.

Proof.

$$(-f)^{*}(\boldsymbol{y}) = \inf_{\boldsymbol{x}} \{\boldsymbol{y}^{T}\boldsymbol{x} \quad (-f)(\boldsymbol{x})\} = \quad \sup_{\boldsymbol{x}} \{-\boldsymbol{y}^{T}\boldsymbol{x} \quad f(\boldsymbol{x})\} = \quad -f^{*}(-\boldsymbol{y})$$

□

A.1.4 Duality theorem

Theorem 13 (Fenchel duality)**.** *Assume $f : \mathsf{R}^n \to \overline{\mathsf{R}}$ is proper and convex and let $g : \mathsf{R}^n \to \overline{\mathsf{R}}$ be a given proper concave function. If the following Slater's conditions hold,*

$$\exists \hat{\boldsymbol{x}} \in \operatorname{int}(\operatorname{dom}(f)) \setminus \operatorname{int}(\operatorname{dom}(-g)) \tag{A.4}$$

then the following equality holds.

$$\inf_{\boldsymbol{x} \in \operatorname{dom}(f) \setminus \operatorname{dom}(g)} \{f(\boldsymbol{x}) - g(\boldsymbol{x})\} = \sup_{\boldsymbol{y} \in \operatorname{dom}(f^*) \setminus \operatorname{dom}(-g^*)} \{g_*(\boldsymbol{y}) - f_*(\boldsymbol{y})\} \tag{A.5}$$

Corrolary 7 (Maximizing a concave function over a convex set)**.** *Let $X \subseteq \mathsf{R}^n$ be a convex set with non-empty interior and let $g : \mathsf{R}^n \to \mathsf{R}$ be a proper concave function over X, then the following holds.*

$$\sup_{\boldsymbol{x} \in X} g(\boldsymbol{x}) = \inf_{\boldsymbol{y} \in dom(-g)} \{\delta^*(\boldsymbol{y} | X) - g_*(\boldsymbol{y})\}$$

A.2 Convex-hull splitting property

The following theorem is derived, and extended, from Arslan and Detienne [2021].

Theorem 14. *Let $Y \subseteq \Pi_{j=1}^n [l_j, u_j]$ and let $L(\boldsymbol{x})$ be de ned, for $\boldsymbol{x} \in \{0,1\}^n$ as follows,*

$$L(\boldsymbol{x}) = \left\{ \boldsymbol{y} \in \mathsf{R}^n_+ : \forall j \in \{1,...,n\}, \begin{cases} x_j = 1 \implies y_j \in [\alpha_j^1, \beta_j^1] \\ x_j = 0 \implies y_j \in [\alpha_j^0, \beta_j^0] \end{cases} \right\} \tag{A.6}$$

with $\alpha_j^0, \alpha_j^1, \beta_j^0, \beta_j^1 \in \{l_j, u_j\}$. Then, the following equality holds,

$$\forall \boldsymbol{x} \in \{0,1\}^n, \quad \operatorname{conv}(Y \setminus L(\boldsymbol{x})) = \operatorname{conv}(Y) \setminus L(\boldsymbol{x}) \tag{A.7}$$

Proof. First, it is clear that $\operatorname{conv}(Y \setminus L(\boldsymbol{x})) \subseteq \operatorname{conv}(Y) \setminus L(\boldsymbol{x})$ for any $\boldsymbol{x} \in \{0,1\}^n$. Thus, assume that there exists $\alpha_1, ..., \alpha_{n+1} \geq 0$ and $\bar{\mathbf{y}}^1, ..., \bar{\mathbf{y}}^{n+1} \in Y$ such that $\sum_{k=1}^{n+1} \alpha_k = 1$ and $\boldsymbol{y} = \sum_{k=1}^{n+1} \alpha_k \bar{\mathbf{y}}^k \in \operatorname{conv}(Y) \setminus L(\boldsymbol{x})$ while $\boldsymbol{y} \notin \operatorname{conv}(Y \setminus L(\boldsymbol{x}))$. Thus, there exists some $\bar{k}$ and $\bar{j}$ such that $\alpha_{\bar{k}} > 0$, $\bar{\mathbf{y}}^{\bar{k}} \in Y$ and $((x_{\bar{j}} = 1) \wedge (\bar{\mathbf{y}}^{\bar{k}}_{\bar{j}} \notin [\alpha^1_{\bar{j}}, \beta^1_{\bar{j}}])) \vee ((x_{\bar{j}} = 0) \wedge (\bar{\mathbf{y}}^{\bar{k}}_{\bar{j}} \notin [\alpha^0_{\bar{j}}, \beta^0_{\bar{j}}]))$.

We only treat the case where $x_{\bar{j}} = 1$, since the case $x_{\bar{j}} = 0$ can be treated similarly. We then have four possibilities:

- $\alpha^1_{\bar{j}} = l_{\bar{j}}$, $\beta^1_{\bar{j}} = l_{\bar{j}}$: Then, $\bar{\mathbf{y}}^{\bar{k}}_{\bar{j}} \notin [\alpha^1_{\bar{j}}, \beta^1_{\bar{j}}]$ implies that $\bar{\mathbf{y}}^{\bar{k}}_{\bar{j}} > l_{\bar{j}}$ and since $\boldsymbol{y} \in L(\boldsymbol{x})$ we must have $y_{\bar{j}} = l_{\bar{j}}$. Then, we can write the following,

$$l_{\bar{j}} = y_{\bar{j}} = \sum_{k=1}^{n+1} \alpha_k \bar{\mathbf{y}}^k_{\bar{j}} > \sum_{k : \bar{\mathbf{y}}^k_{\bar{j}} \in [\alpha^1_{\bar{j}}, \beta^1_{\bar{j}}]} \alpha_k l_{\bar{j}} + \sum_{k : \bar{\mathbf{y}}^k_{\bar{j}} \notin [\alpha^1_{\bar{j}}, \beta^1_{\bar{j}}]} \alpha_k l_{\bar{j}} = l_{\bar{j}} \tag{A.8}$$

  which is absurd.

- $\alpha^1_{\bar{j}} = u_{\bar{j}}$, $\beta^1_{\bar{j}} = l_{\bar{j}}$: Impossible, unless $u_{\bar{j}} = l_{\bar{j}}$

– $\alpha^1_{\bar{j}} = l_{\bar{j}}$, $\beta^1_{\bar{j}} = u_{\bar{j}}$: Then, $\bar{y}^{\bar{k}}_{\bar{j}} \not\in [\alpha^1_{\bar{j}}, \beta^1_{\bar{j}}]$ implies $\bar{\mathrm{y}}^{\bar{k}} \not\in Y$, which violates the assumption.

– $\alpha^1_{\bar{j}} = u_{\bar{j}}$, $\beta^1_{\bar{j}} = u_{\bar{j}}$: this case yields a contradiction with the same argument as in the first case.

□

Remark 19. *Special cases of Theorem 14 are* $L(\boldsymbol{x}) = \{\boldsymbol{y} : \boldsymbol{y} \quad \boldsymbol{x}\}$, $L(\boldsymbol{x}) = \{\boldsymbol{y} : \boldsymbol{y} \quad \boldsymbol{x}\}$, $L(\boldsymbol{x}) = \{\boldsymbol{y} : \boldsymbol{y} = \boldsymbol{x}\}$ *and* $L(\boldsymbol{x}) = \{\boldsymbol{y} : y_j \quad (1 \quad x_j)u_j\}$ *for binary* $\boldsymbol{x}$.

APPENDIX B

Additional proofs

B.1 Proof of Example 11

Assume that g_i $(i = 0, 1, ..., m)$ is generically defined by means of ℓ_p-norms, i.e., assume that $g_i(\boldsymbol{x}, \boldsymbol{y}) = \left|\left|\boldsymbol{K}_X^i \boldsymbol{x} + \boldsymbol{K}_Y^i \boldsymbol{y} + \boldsymbol{\chi}^i\right|\right|_{p_i} + {\boldsymbol{\delta}_X^i}^T \boldsymbol{x} + {\boldsymbol{\delta}_Y^i}^T \boldsymbol{y} + \kappa^i$ where $\boldsymbol{K}_X^i, \boldsymbol{K}_Y^i, \boldsymbol{\chi}^i, \boldsymbol{\delta}_X^i, \boldsymbol{\delta}_Y^i$ and κ^i are given.

B.1.1 Computing convex conjugates

First, observe that

$$g_i \boldsymbol{j}_{\boldsymbol{x}}(\boldsymbol{y}) = h_1(\boldsymbol{y}) + {\boldsymbol{\delta}_X^i}^T \boldsymbol{x} + {\boldsymbol{\delta}_Y^i}^T \boldsymbol{y} + \kappa^i \tag{B.1}$$

where $h_1(\boldsymbol{y}) = \left|\left|\boldsymbol{K}_X^i \boldsymbol{x} + \boldsymbol{K}_Y^i \boldsymbol{y} + \boldsymbol{\chi}^i\right|\right|_{p_i}$. By addition to an affine function, we have

$$g_i \boldsymbol{j}_{\boldsymbol{x}}(\boldsymbol{\pi}) = h_1(\boldsymbol{\pi} \quad \boldsymbol{\delta}_Y^i) \quad {\boldsymbol{\delta}_X^i}^T \boldsymbol{x} \quad \kappa^i. \tag{B.2}$$

Now, we may write h_1 as

$$h_1(\boldsymbol{y}) = h_2(\boldsymbol{K}_Y^i \boldsymbol{y}) \tag{B.3}$$

where $h_2(\boldsymbol{y}) = \left|\left|\boldsymbol{K}_X^i \boldsymbol{x} + \boldsymbol{y} + \boldsymbol{\chi}^i\right|\right|_{p_i}$. By composition with a linear mapping (see Ben-Tal et al. [2014], Lemma 6.7) and since $\mathrm{dom}\,(h_2) = \mathsf{R}^{n_Y}$, we have

$$h_1(\boldsymbol{\pi}) = \inf_{\boldsymbol{\omega}} \{h_2(\boldsymbol{\omega}) : {\boldsymbol{K}_Y^i}^T \boldsymbol{\omega} = \boldsymbol{\pi}\}. \tag{B.4}$$

Together with (B.2), we have

$$g_i \boldsymbol{j}_{\boldsymbol{x}}(\boldsymbol{\pi}) = \inf_{\boldsymbol{\omega}} \{h_2(\boldsymbol{\omega}) : {\boldsymbol{K}_Y^i}^T \boldsymbol{\omega} = \boldsymbol{\pi} \quad \boldsymbol{\delta}_Y^i \} \quad {\boldsymbol{\delta}_X^i}^T \boldsymbol{x} \quad \kappa^i. \tag{B.5}$$

Then,

$$h_2(\boldsymbol{y}) = h_3(\boldsymbol{y} + \boldsymbol{K}_X^i \boldsymbol{x} + \boldsymbol{\chi}^i) \tag{B.6}$$

with $h_3(\boldsymbol{y}) = \boldsymbol{jyjj}_{p_i}$. Thus, by translation of argument, we have

$$h_2(\boldsymbol{\pi}) = h_3(\boldsymbol{\pi}) \quad (\boldsymbol{K}_X^i \boldsymbol{x} + \boldsymbol{\chi}^i)^T \boldsymbol{\pi}. \tag{B.7}$$

Now, h_3 being a norm, its convex conjugate is the indicator of the unit ball for the dual norm, thus,

$$h_3(\boldsymbol{\pi}) = \delta(\boldsymbol{\pi} \boldsymbol{j} B_{p_i^0}(\boldsymbol{0}, 1)) \tag{B.8}$$

with $1/p_i + 1/p_i^0 = 1$. Together with (B.5) and (B.7), we have

$$g_i \boldsymbol{j}_{\boldsymbol{x}}(\boldsymbol{\pi}) = \inf_{\boldsymbol{\omega}} \boldsymbol{f} \delta(\boldsymbol{\omega} \boldsymbol{j} B_{p_i^0}(\boldsymbol{0}, 1)) \quad (\boldsymbol{K}_X^i \boldsymbol{x} + \boldsymbol{\chi}^i)^T \boldsymbol{\omega} : \boldsymbol{K}_Y^{i\,T} \boldsymbol{\omega} = \boldsymbol{\pi} \quad \boldsymbol{\delta}_Y^i \boldsymbol{g}$$
$$\boldsymbol{\delta}_X^{i\,T} \boldsymbol{x} \quad \kappa^i. \tag{B.9}$$

By optimality, we get

$$g_i \boldsymbol{j}_{\boldsymbol{x}}(\boldsymbol{\pi}) = \inf \quad (\boldsymbol{K}_X^i \boldsymbol{x} + \boldsymbol{\chi}^i)^T \boldsymbol{\omega} \quad \boldsymbol{\delta}_X^{i\,T} \boldsymbol{x} \quad \kappa^i \tag{B.10}$$
$$\text{s.t. } \boldsymbol{K}_Y^{i\,T} \boldsymbol{\omega} = \boldsymbol{\pi} \quad \boldsymbol{\delta}_Y^i \tag{B.11}$$
$$\boldsymbol{jj\omega jj}_{p_i^0} \quad 1 \tag{B.12}$$
$$\boldsymbol{\omega}\ \boldsymbol{2}\ \mathsf{R}^{n_Y} \tag{B.13}$$

B.1.2 Applying Theorem 8

By substitution, we get

$$\lambda_i g_i \boldsymbol{j}_{\boldsymbol{x}}(\boldsymbol{u}^i/\lambda_i) = \inf \lambda_i \Big(\quad (\boldsymbol{K}_X^i \boldsymbol{x} + \boldsymbol{\chi}^i)^T \boldsymbol{\omega}^i \quad \boldsymbol{\delta}_X^{i\,T} \boldsymbol{x} \quad \kappa^i \Big) \tag{B.14}$$
$$\text{s.t. } \boldsymbol{K}_Y^{i\,T} \boldsymbol{\omega}^i = \boldsymbol{u}^i/\lambda_i \quad \boldsymbol{\delta}_Y^i \tag{B.15}$$
$$\boldsymbol{jj\omega jj}_{p_i^0} \quad 1 \tag{B.16}$$
$$\boldsymbol{\omega}^i\ \boldsymbol{2}\ \mathsf{R}^{n_Y}. \tag{B.17}$$

Introducing $\boldsymbol{z}^i = \lambda_i \boldsymbol{\omega}^i$, we have

$$\lambda_i g_i \boldsymbol{j}_{\boldsymbol{x}}(\boldsymbol{u}^i/\lambda_i) = \inf \quad (\boldsymbol{K}_X^i \boldsymbol{x} + \boldsymbol{\chi}^i)^T \boldsymbol{z}^i \quad \lambda_i (\boldsymbol{\delta}_X^{i\,T} \boldsymbol{x} + \kappa^i) \tag{B.18}$$
$$\text{s.t. } \boldsymbol{K}_Y^{i\,T} \boldsymbol{z}^i = \boldsymbol{u}^i \quad \lambda_i \boldsymbol{\delta}_Y^i \tag{B.19}$$
$$\big|\big|\boldsymbol{z}^i\big|\big|_{p_i^0} \quad \lambda_i \tag{B.20}$$
$$\boldsymbol{z}^i\ \boldsymbol{2}\ \mathsf{R}^{n_Y}. \tag{B.21}$$

We therefore obtain

$$\sup \sum_{i=0}^{m} \Big((\boldsymbol{K}_X^i \boldsymbol{x} + \boldsymbol{\chi}^i)^T \boldsymbol{z}^i + \lambda_i (\boldsymbol{\delta}_X^{i\,T} \boldsymbol{x} + \kappa^i) \Big) + \boldsymbol{\lambda}^T \boldsymbol{F}(\boldsymbol{x}) \boldsymbol{\xi} \quad \lambda_0 x_0 \tag{B.22}$$
$$\text{s.t. } \sum_{i=0}^{m} \Big(\boldsymbol{K}_Y^{i\,T} \boldsymbol{z}^i + \lambda_i \boldsymbol{\delta}_Y^i \Big) = \boldsymbol{0} \tag{B.23}$$

$$\left|\left|\boldsymbol{z}^i\right|\right|_{p_i^0} \quad \lambda_i \qquad i = 0, 1, ..., m \tag{B.24}$$

$$\boldsymbol{z}^i \; 2 \; \mathsf{R}^{n_Y} \qquad i = 0, 1, ..., m \tag{B.25}$$

$$(\lambda_0, \boldsymbol{\lambda}) \; 2 \; \Lambda \tag{B.26}$$

$$\boldsymbol{\xi} \; 2 \; \Xi. \tag{B.27}$$

By letting $\boldsymbol{a}^i(\boldsymbol{x}) = \boldsymbol{K}_X^i \boldsymbol{x} + \boldsymbol{\chi}^i$, $\boldsymbol{b}(\boldsymbol{x}) = ({\boldsymbol{\delta}_X^1}^T \boldsymbol{x} + \kappa^1, \ldots, {\boldsymbol{\delta}_X^m}^T \boldsymbol{x} + \kappa^m)^T$ and $\boldsymbol{\Delta} = (\boldsymbol{\delta}_Y^1, \ldots, \boldsymbol{\delta}_Y^m)$, we may rewrite it as follows.

$$\sup \sum_{i=0}^{m} \boldsymbol{a}^i(\boldsymbol{x})^T \boldsymbol{z}^i + \left(\boldsymbol{b}(\boldsymbol{x}) + \boldsymbol{F}(\boldsymbol{x})\boldsymbol{\xi}\right)^T \boldsymbol{\lambda} \quad \lambda_0 x_0 \tag{B.28}$$

$$\text{s.t.} \sum_{i=0}^{m} {\boldsymbol{K}_Y^i}^T \boldsymbol{z}^i + \boldsymbol{\Delta}\boldsymbol{\lambda} + \boldsymbol{\delta}_Y^0 \lambda_0 = \boldsymbol{0} \tag{B.29}$$

$$\left|\left|\boldsymbol{z}^i\right|\right|_{p_i^0} \quad \lambda_i \qquad i = 0, 1, ..., m \tag{B.30}$$

$$\boldsymbol{z}^i \; 2 \; \mathsf{R}^{n_Y} \qquad i = 0, 1, ..., m \tag{B.31}$$

$$(\lambda_0, \boldsymbol{\lambda}) \; 2 \; \Lambda \tag{B.32}$$

$$\boldsymbol{\xi} \; 2 \; \Xi \tag{B.33}$$

Bibliography

Ali Allahverdi and John Mittenthal. Scheduling on a two-machine flowshop subject to random breakdowns with a makespan objective function. *European Journal of Operational Research*, 81(2):376 – 387, 1995. ISSN 0377-2217. doi: https://doi.org/10.1016/0377-2217(93)E0318-R. URL http://www.sciencedirect.com/science/article/pii/0377221793E0318R.

Mohamed Ali Aloulou and Federico Della Croce. Complexity of single machine scheduling problems under scenario-based uncertainty. *Oper. Res. Lett.*, 36(3):338–342, May 2008. ISSN 0167-6377. doi: 10.1016/j.orl.2007.11.005. URL http://dx.doi.org/10.1016/j.orl.2007.11.005.

Ayşe N Arslan and Boris Detienne. Decomposition-based approaches for a class of two-stage robust binary optimization problems. *INFORMS Journal on Computing*, 2021. URL https://hal.inria.fr/hal-02190059.

Josette Ayoub and Michael Poss. Decomposition for adjustable robust linear optimization subject to uncertainty polytope. *Computational Management Science*, 13(2):219–239, February 2016. doi: 10.1007/s10287-016-0249-2. URL https://doi.org/10.1007/s10287-016-0249-2.

Philippe Baptiste, Laurent Peridy, and Eric Pinson. A branch and bound to minimize the number of late jobs on a single machine with release time constraints. *European Journal of Operational Research*, 144(1):1 – 11, 2003. ISSN 0377-2217. doi: https://doi.org/10.1016/S0377-2217(01)00353-8. URL http://www.sciencedirect.com/science/article/pii/S0377221701003538.

Aharon Ben-Tal, Dick den Hertog, and Jean-Philippe Vial. Deriving robust counterparts of nonlinear uncertain inequalities. *Mathematical Programming*, 149(1-2):265–299, February 2014. doi: 10.1007/s10107-014-0750-8. URL https://doi.org/10.1007/s10107-014-0750-8.

Pascale Bendotti, Philippe Chrétienne, Pierre Fouilhoux, and Alain Quilliot. Anchored reactive and proactive solutions to the cpm-scheduling problem. *European Journal of Operational Research*, 261(1):67 – 74, 2017. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2017.02.007. URL http://www.sciencedirect.com/science/article/pii/S0377221717301121.

D. Bertsimas and A. Georghiou. Binary decision rules for multistage adaptive mixed-integer optimization. *Mathematical Programming*, 167(2):395–433, March 2017.

Dimitris Bertsimas and Frans J. C. T. de Ruiter. Duality in two-stage adaptive linear optimization: Faster computation and stronger bounds. *INFORMS Journal on Computing*, 28(3):500–511, July 2016. doi: 10.1287/ijoc.2016.0689. URL https://doi.org/10.1287/ijoc.2016.0689.

Dimitris Bertsimas and Melvyn Sim. The price of robustness. *Operations Research*, 52(1):35–53, February 2004. doi: 10.1287/opre.1030.0065. URL https://doi.org/10.1287/opre.1030.0065.

Dimitris Bertsimas, Ebrahim Nasrabadi, and Sebastian Stiller. Robust and Adaptive Network Flows. *Operations Research*, 61(5):1218–1242, October 2013. ISSN 0030-364X, 1526-5463. doi: 10.1287/opre.2013.1200. URL http://pubsonline.informs.org/doi/abs/10.1287/opre.2013.1200.

J. Birge, J. B. G. Frenk, J. Mittenthal, and A. H. G. Rinnooy Kan. Single-machine scheduling subject to stochastic breakdowns. *Naval Research Logistics (NRL)*, 37(5):661–677, 1990.

Odellia Boni and Aharon Ben-Tal. Adjustable robust counterpart of conic quadratic problems. *Mathematical Methods of Operations Research*, 68(2):211–233, April 2008. doi: 10.1007/s00186-008-0218-9. URL https://doi.org/10.1007/s00186-008-0218-9.

Marin Bougeret, Artur Pessoa, and Michael Poss. Robust scheduling with budgeted uncertainty. *Discrete Applied Mathematics*, 08 2018. doi: 10.1016/j.dam.2018.07.001.

C Buchheim and J. Kurtz. Robust combinatorial optimization under convex and discrete cost uncertainty. *EURO Journal on Computational Optimization*, 6(3):211–238, 2018.

A. Caprara, M. Carvalho, A. Lodi, and G. J. Woeginger. A complexity and approximability study of the bilevel knapsack problem. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 98–109. Springer, 2013.

Sebastián Ceria and João Soares. Convex programming for disjunctive convex optimization. *Mathematical Programming*, 86(3):595–614, Dec 1999. ISSN 1436-4646. doi: 10.1007/s101070050106. URL https://doi.org/10.1007/s101070050106.

Matthias Claus and Maximilian Simmoteit. A note on σ_2^p-completeness of a robust binary linear program with binary uncertainty set. *Operations Research Letters*, 48(5):594–598, September 2020. doi: 10.1016/j.orl.2020.07.006. URL https://doi.org/10.1016/j.orl.2020.07.006.

François Clautiaux, Boris Detienne, and Henri Lefebvre. A two-stage robust approach for minimizing the weighted number of tardy jobs with objective uncertainty. *Journal of Scheduling*, February 2023. doi: 10.1007/s10951-022-00775-1. URL https://doi.org/10.1007/s10951-022-00775-1.

G. Cornuéjols, R. Sridharan, and J-M. Thizy. A comparison of heuristics and relaxations for the capacitated plant location problem. *European journal of operational research*, 50(3):280–297, 1991.

George B. Dantzig. Linear programming under uncertainty. *Management Science*, 1(3/4):197–206, 1955. ISSN 00251909, 15265501. URL http://www.jstor.org/stable/2627159.

Stéphane Dauzère-Pérès and Marc Sevaux. Using lagrangean relaxation to minimize the weighted number of late jobs. *Naval Research Logistics*, 50(3):273–288, 2003. doi: 10.1002/nav.10056. URL https://hal.archives-ouvertes.fr/hal-00069413.

Stéphane Dauzère-Pérès. Minimizing late jobs in the general one machine scheduling problem. *European Journal of Operational Research*, 81(1):134 – 142, 1995. ISSN 0377-2217. doi: https://doi.org/10.1016/0377-2217(94)00116-T. URL http://www.sciencedirect.com/science/article/pii/037722179400116T.

Frans J. C. T. de Ruiter, Jianzhe Zhen, and Dick den Hertog. Dual approach for two-stage robust nonlinear optimization. *Operations Research*, April 2022. doi: 10.1287/opre.2022.2289. URL https://doi.org/10.1287/opre.2022.2289.

Erick Delage and Yinyu Ye. Distributionally robust optimization under moment uncertainty with application to data-driven problems. *Operations Research*, 58(3):595–612, June 2010. doi: 10.1287/opre.1090.0741. URL https://doi.org/10.1287/opre.1090.0741.

Martin Desrochers, Patrice Marcotte, and Mihnea Stan. The congested facility location problem. *Location Science*, 3(1):9–23, May 1995. doi: 10.1016/0966-8349(95)00004-2. URL https://doi.org/10.1016/0966-8349(95)00004-2.

B. Detienne, H. Lefebvre, E. Malaguti, and M. Monaci. Adaptive robust optimization with objective uncertainty. *Technical Report OR-21-1, DEI-University of Bologna*, 2021.

Boris Detienne. A mixed integer linear programming approach to minimize the number of late jobs with and without machine availability constraints. *European Journal of Operational Research*, 235:540–552, 2014.

Elizabeth D. Dolan and Jorge J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213, Jan 2002. ISSN 1436-4646. doi: 10.1007/s101070100263. URL https://doi.org/10.1007/s101070100263.

M. Fischetti, I. Ljubić, M. Monaci, and M. Sinnl. Interdiction games and monotonicity, with application to knapsack problems. *INFORMS Journal on Computing*, 31(2):390–410, 2019.

Matteo Fischetti, Ivana Ljubić, and Markus Sinnl. Benders decomposition without separability: A computational study for capacitated facility location problems. *European Journal of Operational Research*, 253(3):557–569, September 2016. doi: 10.1016/j.ejor.2016.03.002. URL https://doi.org/10.1016/j.ejor.2016.03.002.

A. Garcia-Piquer, J. C. Morales, I. Ribas, J. Colomé, J. Guàrdia, M. Perger, J. A. Caballero, M. Cortés-Contreras, S. V. Jeffers, A. Reiners, P. J. Amado, A. Quirrenbach, and W. Seifert. Efficient scheduling of astronomical observations - Application to the CARMENES radial-velocity survey. *Astronomy & Astrophysics*, 604:A87, August 2017. ISSN 0004-6361, 1432-0746. doi: 10.1051/0004-6361/201628577. URL https://www.aanda.org/articles/aa/abs/2017/08/aa28577-16/aa28577-16.html. Publisher: EDP Sciences.

Ricardo García, Angel Marín, and Michael Patriksson. Column generation algorithms for nonlinear optimization, i: Convergence analysis. *Optimization*, 52(2):171–200, 2003. doi: 10.1080/0233193031000079856. URL https://doi.org/10.1080/02331930310000798 56.

A. M. Geoffrion. Generalized benders decomposition. *Journal of Optimization Theory and Applications*, 10(4):237–260, October 1972. doi: 10.1007/bf00934810. URL https://doi.org/10.1007/bf00934810.

Angelos Georghiou, Angelos Tsoukalas, and Wolfram Wiesemann. Robust Dual Dynamic Programming. *Operations Research*, April 2019. doi: 10.1287/opre.2018.1835. URL https://pubsonline.informs.org/doi/abs/10.1287/opre.2018.1835. Publisher: INFORMS.

R.L. Graham, E.L. Lawler, J.K. Lenstra, and A.H.G.Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. In P.L. Hammer, E.L. Johnson, and B.H. Korte, editors, *Discrete Optimization II*, volume 5 of *Annals of Discrete Mathematics*, pages 287 – 326. Elsevier, 1979. doi: https://doi.org/10.1016/S0167-5060(08)70356-X. URL http://www.sciencedirect.com/science/article/pii/S016750600870356X.

Ignacio E. Grossmann and Juan P. Ruiz. Generalized disjunctive programming: A framework for formulation and alternative algorithms for MINLP optimization. In *Mixed Integer Nonlinear Programming*, pages 93–115. Springer New York, November 2011. doi: 10.1007/978-1-4614-1927-3_4. URL https://doi.org/10.1007/978-1-4614-1927-3_4.

Grani A. Hanasusanto, Daniel Kuhn, and Wolfram Wiesemann. K-adaptability in two-stage robust binary programming. *Operations Research*, 63(4):877–891, August 2015. doi: 10.1287/opre.2015.1392. URL https://doi.org/10.1287/opre.2015.1392.

J.R. Jackson. *Scheduling a production line to minimize maximum tardiness*. Research report. Office of Technical Services, 1955.

Nicolas Kämmerling and Jannis Kurtz. Oracle-based algorithms for binary two-stage robust optimization. *Computational Optimization and Applications*, 77(2):539–569, June 2020. doi: 10.1007/s10589-020-00207-w. URL https://doi.org/10.1007/s10589-020-00207-w.

Hiroshi Kise, Toshihide Ibaraki, and Hisashi Mine. A solvable case of the one-machine scheduling problem with ready and due times. *Oper. Res.*, 26(1):121–126, February 1978. ISSN 0030-364X. doi: 10.1287/opre.26.1.121. URL http://dx.doi.org/10.1287/opre.26.1.121.

Nikolaos H. Lappas and Chrysanthos E. Gounaris. Multi-stage adjustable robust optimization for process scheduling under uncertainty. *AIChE Journal*, 62(5):1646–1667, 2016. ISSN 1547-5905. doi: 10.1002/aic.15183. URL https://aiche.onlinelibrary.wiley.com/doi/abs/10.1002/aic.15183. _eprint: https://aiche.onlinelibrary.wiley.com/doi/pdf/10.1002/aic.15183.

Can Li and Ignacio E. Grossmann. A finite ε-convergence algorithm for two-stage stochastic convex nonlinear programs with mixed-binary first and second-stage variables. *J. Glob. Optim.*, 75(4):921–947, 2019. doi: 10.1007/s10898-019-00820-y. URL https://doi.org/10.1007/s10898-019-00820-y.

Zukui Li, Ran Ding, and Christodoulos A. Floudas. A comparative theoretical and computational study on robust counterpart optimization: I. robust linear optimization and robust mixed integer linear optimization. *Industrial and Engineering Chemistry Research*, 50(18):10567–10603, August 2011. doi: 10.1021/ie200150p. URL https://doi.org/10.1021/ie200150p.

Fengming Lin, Xiaolei Fang, and Zheming Gao. Distributionally robust optimization: A review on theory and applications. *Numerical Algebra, Control and Optimization*, 12(1):159, 2022. doi: 10.3934/naco.2021057. URL https://doi.org/10.3934/naco.2021057.

James Luedtke. An integer programming and decomposition approach to general chance-constrained mathematical programs. In *Integer Programming and Combinatorial Optimization*, pages 271–284. Springer Berlin Heidelberg, 2010. doi: https://10.1007/978-3-642-13036-6*n*_21. URL https://doi.org/10.1007/978-3-642-13036-6_21.

Jyotsna K. Mandal, Somnath Mukhopadhyay, and Paramartha Dutta, editors. *Multi-Objective Optimization*. Springer Singapore, 2018. doi: 10.1007/978-981-13-1471-1. URL https://doi.org/10.1007/978-981-13-1471-1.

Ahmadreza Marandi and Dick den Hertog. When are static and adjustable robust optimization problems with constraint-wise uncertainty equivalent? *Mathematical Programming*, 170 (2):555–568, June 2017. doi: 10.1007/s10107-017-1166-z. URL https://doi.org/10.1007/s10107-017-1166-z.

Rym M'Hallah and R.L. Bulfin. Minimizing the weighted number of tardy jobs on a single machine with release dates. *European Journal of Operational Research*, 176(2):727 – 744, 2007. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2005.08.013. URL http://www.sciencedirect.com/science/article/pii/S0377221705006958.

J v Neumann. Zur theorie der gesellschaftsspiele. *Mathematische annalen*, 100(1):295–320, 1928.

Vianney Perchet and Guillaume Vigeral. A minmax theorem for concave-convex mappings with no regularity assumptions. *Journal of Convex Analysis*, 22, 01 2015.

Artur Pessoa, Ruslan Sadykov, Eduardo Uchoa, and François Vanderbeck. Automation and combination of linear-programming based stabilization techniques in column generation. *INFORMS Journal on Computing*, 30(2):339–360, 2018.

Michael L. Pinedo. *Scheduling: Theory, Algorithms, and Systems*. Springer Publishing Company, Incorporated, 6th edition, 2016. ISBN 0387789340.

Laurent Péridy, Eric Pinson, and David Rivreau. Using short-term memory to minimize the weighted number of late jobs on a single machine. *European Journal of Operational Research*, 148(3):591 – 603, 2003. ISSN 0377-2217. doi: https://doi.org/10.1016/S0377-2217(02)00438-1. URL http://www.sciencedirect.com/science/article/pii/S0377221702004381.

Ralph Tyrell Rockafellar. *Convex analysis*. Princeton Landmarks in Mathematics and Physics. Princeton University Press, Princeton, NJ, December 1996.

Ruslan Sadykov. A branch-and-check algorithm for minimizing the weighted number of late jobs on a single machine with release dates. *European Journal of Operational Research*, 189(3): 1284 – 1304, 2008. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2006.06.078. URL http://www.sciencedirect.com/science/article/pii/S0377221707005991.

Johannes Schneider and Scott Kirkpatrick. *Stochastic Optimization*. Scientific Computation. Springer, Berlin, Germany, November 2010.

Alexander Shapiro. Minimax and risk averse multistage stochastic programming. *European Journal of Operational Research*, 219(3):719–726, June 2012. ISSN 0377-2217. doi: 10.1016/j.ejor.2011.11.005. URL http://www.sciencedirect.com/science/article/pii/S0377221711009921.

Hanif Sherali and Barbara Fraticelli. A modification of benders' decomposition algorithm for discrete subproblems: An approach for stochastic programs with integer recourse. *Journal of Global Optimization*, 22:319–, 01 2002. doi: 10.1023/A:1013827731218.

Hanif D. Sherali and Xiaomei Zhu. On solving discrete two-stage stochastic programs having mixed-integer first- and second-stage variables. *Mathematical Programming*, 108(2):597–616, Sep 2006. ISSN 1436-4646. doi: 10.1007/s10107-006-0724-6. URL https://doi.org/10.1007/s10107-006-0724-6.

Herbert A. Simon. Decision making: Rational, nonrational, and irrational. *Educational Administration Quarterly*, 29(3):392–411, August 1993. doi: 10.1177/0013161x93029003009. URL https://doi.org/10.1177/0013161x93029003009.

Anirudh Subramanyam, Chrysanthos E. Gounaris, and Wolfram Wiesemann. K-adaptability in two-stage mixed-integer robust optimization. *Mathematical Programming Computation*, 12(2): 193–224, November 2019. doi: 10.1007/s12532-019-00174-2. URL https://doi.org/10.1007/s12532-019-00174-2.

A. Takeda, S. Taguchi, and R. H. Tütüncü. Adjustable robust optimization models for a nonlinear two-period system. *Journal of Optimization Theory and Applications*, 136(2):275–295, October 2007. doi: 10.1007/s10957-007-9288-8. URL https://doi.org/10.1007/s10957-007-9288-8.

A. M. Turing. On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London Mathematical Society*, s2-42(1):230–265, 1937. doi: 10.1112/plms/s2-42.1.230. URL https://doi.org/10.1112/plms/s2-42.1.230.

Marjan van den Akker, Han Hoogeveen, and Judith Stoef. Combining two-stage stochastic programming and recoverable robustness to minimize the number of late jobs in the case of uncertain processing times. *Journal of Scheduling*, 21(6):607–617, December 2018. ISSN 1094-6136, 1099-1425. doi: 10.1007/s10951-018-0559-z. URL http://link.springer.com/10.1007/s10951-018-0559-z.

Ruby van Rooyen, Deneys S. Maartens, and Peter Martinez. Autonomous observation scheduling in astronomy. In Alison B. Peck, Robert L. Seaman, and Chris R. Benn, editors, *Observatory Operations: Strategies, Processes, and Systems VII*, volume 10704, pages 393 – 408. International Society for Optics and Photonics, SPIE, 2018. doi: 10.1117/12.2311839. URL https://doi.org/10.1117/12.2311839.

F. Vanderbeck. Bapcod – a generic branch-and-price code, 2005. URL https://realopt.bordeaux.inria.fr/?page_id=2.

Jian Yang and Gang Yu. On the robust single machine scheduling problem. *Journal of Combinatorial Optimization*, 6(1):17–33, Mar 2002. ISSN 1573-2886. doi: 10.1023/A:1013333232691. URL https://doi.org/10.1023/A:1013333232691.

Bo Zeng and Long Zhao. Solving two-stage robust optimization problems using a column-and-constraint generation method. *Operations Research Letters*, 41(5):457–461, September 2013. doi: 10.1016/j.orl.2013.05.003. URL https://doi.org/10.1016/j.orl.2013.05.003.

Long Zhao and Bo Zeng. An exact algorithm for two-stage robust optimization with mixed integer recourse problems. *Technical report, University of South Florida*, 2012.